Optimization Methods and Software
Vol. 00, No. 00, Month 200x, 1–28

On large scale unconstrained optimization problems and higher order methods

GEIR GUNDERSEN* and TROND STEIHAUG
University of Bergen, Department of Informatics, High Technology Center
N-5020 Bergen, Norway

(Received 00 Month 200x; In final form 00 Month 200x)

Third order methods will in most cases use fewer iterations than a second order method to reach the same accuracy. However, the number of arithmetic operations per iteration is higher for third order methods than a second order method. Newton's method is the most commonly used second order method and Halley's method is the most well-known third order method. Newton's method is more used in practical applications than any third order method. We will show that for a large class of problems the ratio of the number of arithmetic operations of Halley's method and Newton's method is constant per iteration. It is shown that

$$\frac{\text{One Step Halley}}{\text{One Step Newton}} \leq 5.$$

We show that the zero elements in the Hessian matrix induce zero elements in the tensor (third derivative). The sparsity structure in the Hessian matrix we consider is the skyline or envelope structure. This is a very convenient structure for solving linear systems of equations with a direct method. The class of matrices that have a skyline structure includes banded and dense matrices. Numerical testing confirm that the ratio of the number of arithmetic operations of a third order method and Newton's method is constant per iteration, and is independent of the number of unknowns.

Keywords: Unconstrained optimization problems, induced sparsity, higher order methods, tensor computations

AMS Subject Classification: 65K05; 65F50; 90C30; 90C06

1 Introduction

Consider the system of nonlinear equations

$$F(x) = 0 \tag{1}$$

*Corresponding author. Telephone: (+47) 55 58 41 24; Fax: (+47) 55 58 41 99; E-mail: Geir.Gundersen@ii.uib.no

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is two times continuously differentiable. Iterative methods for solving (1) generate a sequence of iterates $\{x_k\}$ so that for any x_0 sufficiently close to a solution x_* then $x_k \to x_*$. Newton's method is known to exhibit second Q-order rate of convergence using one function value and the derivative at x_k, i.e. $F(x_k)$ and $F'(x_k)$ per iteration, provided the Jacobian matrix at x_* is nonsingular. One of the best known methods for solving (1) that has local third Q-order rate of convergence is Halley's method. Halley's method uses $F(x_k), F'(x_k)$, and $F''(x_k)$ at every iteration. Assumptions for local convergence and rate of convergence of Halley's method are discussed in Yamamoto [1, 2]. Other methods can be constructed that has a local third Q-order rate of convergence by first computing $F(x_k)$ and $F'(x_k)$ and then computing $F(y_k)$ for a suitable y_k that depends on the previously computed quantities. Modified Newton and two step Newton [3] will use y_k to be the previous computed Newton step. We will only consider methods that for every iteration require the function and its derivatives computed at the same point x_k.

Finding a stationary point of the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is solving a system of equations

$$F(x) \equiv \nabla f(x) = 0$$

where $\nabla f(x)$ is the gradient of f at x provided f is three times continuously differentiable. We will show that the cost of computing the $F'' = \nabla^3 f$ can be considerably reduced using all the symmetries in $\nabla^3 f$.

In section 2 it is shown that the Halley method can be derived from two iterations of Newtons method applied on the second order approximation of $F(x)$. In section 3 we discuss the concept of induced sparsity. We define a skyline matrix and derive the number of nonzero elements in a Hessian and in an induced tensor for dense, banded and skyline structured Hessian matrices. In section 4 we define the basic tensor operations needed for third order methods. Algorithms are given and complexity are shown for the basic tensor operations. In section 5 it is shown how to implement third order methods for unconstrained optimization, in addition we show the complexity for the operations and methods. Thus given the computational requirements we prove that the ratio of the number of arithmetic operations of a third order method and Newton's method is constant per iteration, when the structure of tensor is induced by the structure of the Hessian matrix. Finally in section 6 we present some numerical results.

2 The Halley Class

Gutiérrez and Hernández [4] defined a class of methods for solving the nonlinear system of equations (1). For a given parameter α and a starting point x_0 the iterates are defined by

$$x_{k+1} = x_k - \{I + \frac{1}{2}L(x_k)[I - \alpha L(x_k)]^{-1}\}(F^{'}(x_k))^{-1}F(x_k), \quad k \geq 0 \tag{3}$$

where I is the identity matrix and

$$L(x) = (F^{'}(x))^{-1}F^{''}(x)(F^{'}(x))^{-1}F(x). \tag{4}$$

For $\alpha = 0$ the method is the called Chebyshev's method [2], Halley's method is $\alpha = \frac{1}{2}$ [1, 2, 6], and for $\alpha = 1$ the method is referred to as Super Halley's method [4,8]. The Chebyshev's method was generalized to systems of equations by Nečepuerenko [5] and Halley's method extended to systems of equations by Mertvecova [7].

Schwetlick [9] defines a family of methods parameterized with the real scalar α and iteration i:

$$\begin{aligned} \text{Solve for } y_k^{(i+1)} : 0 = {} & F(x_k) + F^{'}(x_k)(y_k^{(i+1)} - x_k) \\ & + \alpha F^{''}(x_k)(y_k^{(i)} - x_k)(y_k^{(i+1)} - x_k) \\ & + (\frac{1}{2} - \alpha)F^{''}(x_k)(y_k^{(i)} - x_k)(y_k^{(i)} - x_k), i = 0, \ldots \end{aligned} \tag{5}$$

where $y_k^0 = x_k$ and $x_{k+1} = y_k^{(i+1)}$. We will show that for $i = 1$ these two classes of methods are equivalent by showing a relationship to two steps of a minor modification of Newton's method applied to the second order Taylor approximation of the function. This relationship is suitable for deriving efficient implementations of all members in the Halley class.

Consider (3) and eliminate the iterates x_k in F and L, and the iteration index k. One step of a method in the Halley class can be written as

$$x_+ = x - \{I + \frac{1}{2}L(I - \alpha L)^{-1}\}(F^{'})^{-1}F \tag{6}$$

where $L = (F^{'})^{-1}F^{''}(F^{'})^{-1}F$. Define $s^{(1)}$ to be the solution of the linear system

$$F's^{(1)} = -F \tag{7}$$

and let $s^{(2)} = x_+ - (x + s^{(1)})$. Observe that

$$I + \frac{1}{2}L(I - \alpha L)^{-1} = (I - \alpha L)^{-1}(I + (\frac{1}{2} - \alpha)L)$$

then (6) may be written as

$$s^{(1)} + s^{(2)} = (I - \alpha L)^{-1}\left(s^{(1)} + (\frac{1}{2} - \alpha)Ls^{(1)}\right)$$

using (7). Thus $s^{(2)}$ must satisfy the linear system of equations

$$(I - \alpha L)s^{(2)} = \frac{1}{2}Ls^{(1)}.$$

From (4) and multiplying by F' on both sides we have that $s^{(2)}$ is the solution of the linear equation

$$\left(F' + \alpha F''s^{(1)}\right)s^{(2)} = -\frac{1}{2}F''s^{(1)}s^{(1)}.$$

We have thus shown that the Halley class of methods (3) can be written as

$$F'(x_k)s_k^{(1)} = -F(x_k) \tag{8}$$

$$\left(F'(x_k) + \alpha F^{''}(x_k)s_k^{(1)}\right)s_k^{(2)} = -\frac{1}{2}F^{''}(x_k)s_k^{(1)}s_k^{(1)} \tag{9}$$

$$x_{k+1} = x_k + s_k^{(1)} + s_k^{(2)}.$$

Consider the class of methods (5). It follows directly from (5) that $s_k^{(1)} = y_k^{(1)} - x_k$ using that $y_k^{(0)} = x_k$. Hence that $s_k^{(1)}$ is the Newton step (8). Consider the equation for $y_k^{(2)}$ in (5), then

$$\begin{aligned} 0 &= F(x_k) + F^{'}(x_k)(y_k^{(2)} - x_k) + \alpha F^{''}(x_k)s_k^{(1)}(y_k^{(2)} - x_k) \qquad (10)\\ &\quad + (\frac{1}{2} - \alpha)F^{''}(x_k)s_k^{(1)}s_k^{(1)} \\ &= -F'(x_k)s_k^{(1)} + F^{'}(x_k)(y_k^{(2)} - x_k) + \alpha F^{''}(x_k)s_k^{(1)}(y_k^{(2)} - x_k) \qquad (11)\\ &\quad + (\frac{1}{2} - \alpha)F^{''}(x_k)s_k^{(1)}s_k^{(1)}, \end{aligned}$$

by invoking (7). Then

$$\begin{aligned} 0 &= F'(x_k)(y_k^{(2)} - x_k - s_k^{(1)}) + \alpha F''(x_k)s_k^{(1)}(y_k^{(2)} - x_k) \qquad (12)\\ &\quad + \frac{1}{2}F''(x_k)s_k^{(1)}s_k^{(1)} - \alpha F''(x_k)s_k^{(1)}s_k^{(1)} \\ &= F'(x_k)(y_k^{(2)} - x_k - s_k^{(1)}) + \alpha F''(x_k)s_k^{(1)}(y_k^{(2)} - x_k - s_k^{(1)}) \qquad (13)\\ &\quad + \frac{1}{2}F''(x_k)s_k^{(1)}s_k^{(1)}, i = 0, \ldots \end{aligned}$$

and $y_k^{(2)} - x_k - s_k^{(1)}$ must satisfies the equation

$$\left(F'(x_k) + \alpha F''(x_k)s_k^{(1)}\right)(y_k^{(2)} - x_k - s_k^{(1)}) + \frac{1}{2}F''s^{(1)}s^{(1)} = 0, \qquad (14)$$

and we have shown that $s_k^{(2)} = y_k^{(2)} - x_k - s_k^{(1)}$ where $s_k^{(2)}$ is defined in (9). Hence, $x_{k+1} = y_k^{(2)} = x_k + s_k^{(1)} + s_k^{(2)}$. We have thus shown that the class defined by Gutiérrez and Hernández [4] is equivalent to the class defined by Schwetlick [9] for $i = 1$.

Consider the second order approximation to the function F at $x + s$

$$F(x + s) \approx T(s) \equiv F(x) + F'(x)s + \frac{1}{2}F''(x)ss.$$

To solve the quadratic equation $T(s) = 0$ we use two iterations of Newton's method

$$\begin{aligned} T'(s^{(0)})s^{(1)} &= -T(s^{(0)}) \\ T'(s^{(0)} + s^{(1)})s^{(2)} &= -T(s^{(0)} + s^{(1)}) \\ x_+ &= x + s^{(0)} + s^{(1)} + s^{(2)}. \end{aligned}$$

By using the fact that $T'(s) = F'(x) + F''(x)s$ and choosing $s^{(0)} = 0$ we see that $s^{(1)}$ will satisfy $F'(x)s^{(1)} = -F(x)$. Further, $T(s^{(0)}+s^{(1)}) = \frac{1}{2}F''(x)(s^{(0)}+ s^{(1)})(s^{(0)} + s^{(1)})$. Then $s^{(2)}$ will satisfy the equation from (14)

$$\left(F'(x) + F''(x)s^{(1)}\right)s^{(2)} = -\frac{1}{2}F''(x)s^{(1)}s^{(1)}$$

using $s^{(0)} = 0$ and we see that this corresponds to $\alpha = 1$ in (3) and (5) when $i = 1$. Werner [3] made the observation that the second step is a Newton step

(14) on the quadratic equation with starting point $s^{(1)}$.

It can easily be shown by induction that for Schwetlick class (5) with $\alpha = 1$ and $i > 1$ is $i + 1$ Newton iteration on $T(s) = 0$ using $s^{(0)} = 0$.

3 Sparsity

Consider the unconstrained optimization problem (2) where $f : \mathbb{R}^n \to \mathbb{R}$ is three times continuously differentiable. For a given $x \in \mathbb{R}^n$ let

$$H_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \text{ and } \mathcal{T}_{ijk} = \frac{\partial^3 f(x)}{\partial x_i \partial x_j \partial x_k} \quad , 1 \leq i, j, k \leq n, \tag{15}$$

then $H \in \mathbb{R}^{n \times n}$ is a symmetric matrix $H_{ij} = H_{ji}$, $i \neq j$, and $\mathcal{T} \in \mathbb{R}^{n \times n \times n}$ is a super-symmetric tensor

$$\begin{aligned} \mathcal{T}_{ijk} = \mathcal{T}_{ikj} = \mathcal{T}_{jik} = \mathcal{T}_{jki} = \mathcal{T}_{kij} = \mathcal{T}_{kji}, \quad & i \neq j, j \neq k, i \neq k \\ \mathcal{T}_{iik} = \mathcal{T}_{iki} = \mathcal{T}_{kii}, \quad & i \neq k. \end{aligned}$$

Hessian matrices are called sparse if some of the elements are zero for all $x \in \mathbb{R}^n$ [10]. Let

$$\frac{\partial^2}{\partial x_i \partial x_j} f(x) = 0, \quad \forall x \in \mathbb{R}^n,$$

then for this pair of indices (i, j) we have

$$\mathcal{T}_{kij} = \mathcal{T}_{ikj} = \mathcal{T}_{ijk} = \frac{\partial^3}{\partial x_i \partial x_j \partial x_k} f(x) = 0 \text{ for } k = 1, \ldots, n \tag{16}$$

since the function is three times continuously differentiable. We say that sparsity structure of the tensor is induced by the sparsity structure of the Hessian matrix.

For a $n \times n$ symmetric matrix H we only need to store the nonzero elements H_{ij} for which $1 \leq j \leq i \leq n$ when the sparsity structure is known a priori. For a $n \times n \times n$ super-symmetric tensor $\mathcal{T}$ we only need to store the nonzero elements $\mathcal{T}_{ijk}$ where $1 \leq k \leq j \leq i \leq n$ [12]. We will show that the sparsity structure of the tensor can be computed from the sparsity structure of the Hessian matrix. This is illustrated in the Figures 1 and 2, which shows the nonzero elements of a 9×9 arrowhead and tridiagonal matrix, respectively. The black squares are stored non-zero elements, while the white squares are

non-zero elements that need not to be stored. The remaining elements are zero. The induced structure of the tensors of the matrices are shown in Figures 3 and 4. The 'boxes' are the stored nonzero elements of tensors induced by an arrowhead and tridiagonal symmetric matrix where $n = 9$.

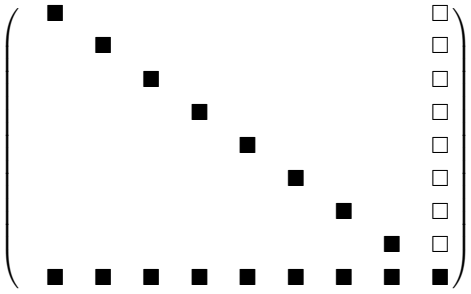

Figure 1. A 9×9 Arrowhead matrix.

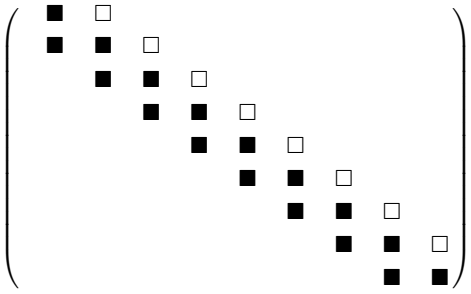

Figure 2. A 9×9 Tridiagonal matrix.

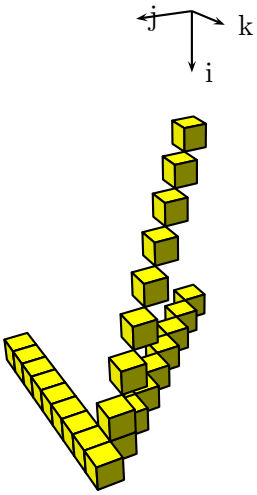

Figure 3. Stored elements of the tensor induced by an arrowhead symmetric matrix for $n = 9$.

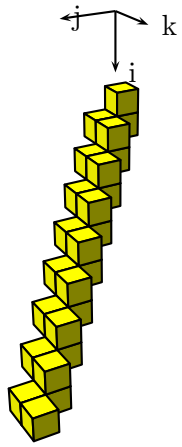

Figure 4. Stored elements of the tensor induced by a tridiagonal symmetric matrix for $n = 9$.

A particular useful sparsity structure of a symmetric matrix is the envelope or skyline structure. A symmetric $n \times n$ matrix H has a skyline structure (envelope structure) if all nonzero elements in a row are located from the first nonzero element to the element on the diagonal. Let f_i be the index of the first nonzero element in row i of the matrix H

$$f_i = 1 + \max\{j |\ H_{ij} = 0,\ j < i\}, \tag{17}$$

assuming that $H_{ii} \neq 0$.

For a Hessian matrix with a skyline structure

$$\frac{\partial^2}{\partial x_i \partial x_j} f(x) = 0, \ \ \forall x \in \mathbb{R}^n, \ \ 1 \leq j < f_i, i = 1, \ldots, n. \tag{18}$$

The following theorem shows that the sparsity structure of the third derivative can be derived from the sparsity of the second derivative and the structure of the tensor is induced by the structure of the Hessian matrix.

THEOREM 3.1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be three times continuously differentiable and assume that the Hessian matrix has a skyline structure (18). The indices of the only possible nonzero elements to be stored of the tensor $\mathcal{T}_{ijk}$ in (15) are the triplets where (i, j, k)*

$$i = 1, \ldots, n, \ \ f_i \leq j \leq i, \ \ \max\{f_i, f_j\} \leq k \leq j.$$

Proof Since $\frac{\partial^2}{\partial x_i \partial x_j} f(x) = 0$ for all x for $j < f_i$ we only need to consider element $\mathcal{T}_{ijk}$ in the tensor so that

$$f_i \leq k \leq j \leq i.$$

Consider the case $f_j > f_i, j \leq i$. Then for k so that $f_i \leq k < f_j$ the tensor element $\mathcal{T}_{ijk} = 0$ since $\frac{\partial^2}{\partial x_j \partial x_k} f(x) = 0$. Hence for i and j so that $f_i \leq j \leq i$

$$\mathcal{T}_{ijk} = 0 \text{ for } k < \max\{f_i, f_j\}.$$

Thus we have shown that the set of indices of the only possible nonzero elements is

$$\{(i, j, k) | 1 \leq i \leq n, \ f_i \leq j \leq i, \ \max\{f_i, f_j\} \leq k \leq j\}.$$

□

We will restrict our attention to tensor induced by the skyline structure of a matrix and all algorithms presented in this paper are designed for this structure.

3.1 *Number of Nonzero Elements*

In this section we derive the number of nonzero elements that needs to be stored in the symmetric Hessian matrix with skyline structure and the induced

tensor. The number of nonzero elements of a sparse matrix is an important number for a data structure and is a measure on the complexity of computing with the matrix and tensor. Since the matrices we consider are all symmetric we only need to consider the lower triangular part of the matrix.

Let $\mathrm{nnz}(H)$ be the number of nonzero elements H_{ij} with indices $1 \leq j \leq i \leq n$ of matrix H. Let f_i be the index of the first nonzero element in row $i, i = 1, \ldots, n$ of H then he number of nonzero elements that need to be stored for a (symmetric) matrix with skyline structure is given by

$$\mathrm{nnz}(H) = \sum_{i=1}^{n} (i - f_i + 1). \tag{19}$$

The number of nonzero elements in a tensor induced by a Hessian matrix with skyline structure follows from Theorem 3.1:

$$\mathrm{nnz}(\mathcal{T}) = \sum_{i=1}^{n} \sum_{j=f_i}^{i} (j - \max\{f_i, f_j\} + 1). \tag{20}$$

If the Hessian matrix has no zero elements, the matrix is dense and $f_i = 1,\ i = 1, \ldots, n$. It follows immediately for a dense symmetric matrix that

$$\mathrm{nnz}(H) = \frac{1}{2} n(n+1). \tag{21}$$

Since, in that case, $\max\{f_i, f_j\} = 1$, it follows from (20) that

$$\mathrm{nnz}(\mathcal{T}) = \sum_{i=1}^{n} \sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{1}{2} i(i+1) = \frac{1}{6} n(n+1)(n+2) \tag{22}$$

The stored elements of a tensor induced from a dense Hessian matrix forms a tetrahedron [12].

It is convenient to introduce the (half) bandwidth of a matrix with skyline structure. Let $\beta_i = i - f_i,\ i = 1, \ldots, n$. If $\beta_i = \beta$ for $i \geq \beta$ the matrix is banded and

$$f_i = \begin{cases} 1 \text{ for } i = 1, \ldots, \beta \\ i - \beta \text{ for } i = \beta + 1, \ldots, n. \end{cases}$$

The number of nonzero elements of a band matrix with constant (half) band-

width β is

$$\text{nnz}(H) = \sum_{i=1}^{\beta} i + \sum_{i=\beta+1}^{n} (\beta+1) = (\beta+1)(n - \frac{\beta}{2}). \tag{23}$$

Consider the tensor induced by a banded Hessian matrix. We note that $\max\{f_i, f_j\} = 1, \quad i = 1, \ldots, \beta$ and $\max\{f_i, f_j\} = i - \beta, \quad i = \beta+1, \ldots, n$ for $j \leq i$. Then from (20) we have

$$\begin{aligned}
\text{nnz}(\mathcal{T}) &= \sum_{i=1}^{\beta} \sum_{j=1}^{i} j + \sum_{i=\beta+1}^{n} \sum_{j=i-\beta}^{i} (j - (i-\beta) + 1) \\
&= \frac{1}{6}\beta(\beta+1)(\beta+2) + \frac{1}{2}(n-\beta)(\beta+1)(\beta+2) \\
&= \frac{1}{2}(\beta+1)(\beta+2)(n - \frac{2}{3}\beta).
\end{aligned} \tag{24}$$

3.2 *Skyline Structure and Partially Separable functions*

An important class of partially separable functions [10] are the so called chained functions. A partially separable function of the form

$$f(x) = \sum_{j=1}^{m} \phi_j(x_{j-\beta_j^{(l)}}, \ldots, x_{j+\beta_j^{(u)}}), \text{ where } 1 \leq j - \beta_j^{(l)} \leq j + \beta_j^{(u)} \leq n \tag{25}$$

is called chained. For $m = n-k+1, \beta_j^{(l)} = 0, \beta_j^{(u)} = k-1, j = 1, \ldots, n-k+1$ we have a k-chained function [13].

In this section we show that a chained partially separable function on the form (25) that is sufficiently smooth has Hessian matrix which has a skyline structure and the Hessian matrix is the sum of small dense blocks centered on the diagonal where each block is the Hessian matrix of an element function ϕ_j.

THEOREM 3.2 *Let $f : \mathbb{R}^n \to \mathbb{R}$ of the form (25) be twice continuously differentiable. Then the Hessian matrix has a skyline structure and $f_i, \ i = 1, 2, \ldots, n$ defined in (17) is monotonically increasing.*

Proof Let J_i be the set of the indices of element functions containing x_i

$$J_i = \{j : j - \beta_j^{(l)} \leq i \leq j + \beta_j^{(u)}\}, \ \ i = 1, 2, \ldots, n.$$

Define

$$k_i^{(l)} = \min_{j \in J_i}\{j - \beta_j^{(l)}\}, \text{ and } k_i^{(u)} = \max_{j \in J_i}\{j + \beta_j^{(u)}\}.$$

and consider element i of the gradient of f

$$\begin{aligned}\frac{\partial f}{\partial x_i}(x) &= \sum_{j=1}^{m} \frac{\partial}{\partial x_i}\phi_j(x_{j-\beta_j^{(l)}}, \ldots, x_{j+\beta_j^{(u)}}) \\ &= \sum_{j \in J_i} \frac{\partial}{\partial x_i}\phi_j(x_{j-\beta_j^{(l)}}, \ldots, x_{j+\beta_j^{(u)}}).\end{aligned}$$

Element i in the gradient depends on

$$x_{k_i^{(l)}}, x_{k_i^{(l)}+1}, \ldots, x_i, \ldots, x_{k_i^{(u)}-1}, x_{k_i^{(u)}}.$$

Hence

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) = 0 \text{ for } j = 1, \ldots, k_i^{(l)} - 1.$$

The index of the first nonzero element in row i is $f_i = k_i^{(l)}$.

To show that f_i is monotonically increasing we proceed by contradiction and assume there exists an index i so that $f_{i+1} < f_i$. Let j' be an element function so that $j' - \beta_{j'}^{(l)} = k_{i+1}^{(l)}$. Then

$$f_{i+1} = k_{i+1}^{(l)} = j' - \beta_{j'}^{(l)} < f_i \leq i.$$

Since $j' - \beta_{j'}^{(l)} < i$ we have $j' \in J_i$ and

$$f_i = k_i^{(l)} = \min_{j \in J_i}\{j - \beta_j^{(l)}\} \leq j' - \beta_{j'}^{(l)} = f_{i+1}.$$

This violates the assumption that $f_{i+1} < f_i$ hence there does not exists i so that $f_{i+1} < f_i$ and we can conclude that f_i is monotonically increasing $i = 1, \ldots, n$. □

Chained partially separable functions have a skyline structure of the Hessian matrix and their tensor has zero elements $\mathcal{T}_{ijk} = 0$ for $1 \leq k < f_i$ for $f_i \leq j \leq i$ since $\max\{f_i, f_j\} = f_i$ in this case. Most of the test examples in [14] are chained functions.

4 Tensor Computations

In this section we present the basic tensor computations needed for the Halley class of methods for solving the unconstrained optimization problem (2). We will use the notation $(p\mathcal{T})$ for the symmetric $n \times n$ matrix $\nabla^3 f(x)p$, where $\mathcal{T}$ is defined in (15). The two quantities needed are $(p\mathcal{T})p \in \mathbb{R}^n$ and $(p\mathcal{T}) \in \mathbb{R}^{n\times n}$.

However, we will illustrate the use of the super-symmetry and sparsity by considering the scalar term $p^T(p\mathcal{T})p$. The scalar term is defined as

$$p^T(p\mathcal{T})p = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} p_i p_j p_k \mathcal{T}_{ijk}. \tag{26}$$

By taking into account that the only elements $\mathcal{T}_{ijk}$ stored will be for $1 \le k \le j \le i \le n$, we have the following

$$\begin{aligned} p^T(p\mathcal{T})p &= \sum_{i=1}^{n} p_i \sum_{j=1}^{n} p_j \sum_{k=1}^{n} p_k \mathcal{T}_{ijk} \\ &= \sum_{i=1}^{n} p_i \left\{ \sum_{j=1}^{i-1} p_j \left(6\sum_{k=1}^{j-1} p_k \mathcal{T}_{ijk} + 3p_j \mathcal{T}_{ijj} \right) + p_i \left(3\sum_{k=1}^{i-1} p_k \mathcal{T}_{iik} + p_i \mathcal{T}_{iii} \right) \right\} \end{aligned}$$

Using that the tensor is induced by a skyline structured matrix and invoking Theorem 3.1 we have

$$p^T(p\mathcal{T})p = \sum_{i=1}^{n} p_i \left\{ \sum_{j=f_i}^{i-1} p_j \left(6\sum_{k=\max\{f_i,f_j\}}^{j-1} p_k \mathcal{T}_{ijk} + 3p_j \mathcal{T}_{ijj} \right) + p_i \left(3\sum_{k=f_i}^{i-1} p_k \mathcal{T}_{iik} + p_i \mathcal{T}_{iii} \right) \right\}$$

To get the number of arithmetic operations for computing $p^T(p\mathcal{T})p$ we first define

$$\sigma_3(i,j) = \sum_{k=\max\{f_i,f_j\}}^{j-1} p_k \mathcal{T}_{ijk}, \quad \sigma_2(i) = \sum_{k=f_i}^{i-1} p_k \mathcal{T}_{iik},$$

and

$$\sigma_1(i) = 3 \sum_{j=f_i}^{i-1} p_j(2\sigma_3(i,j) + p_j \mathcal{T}_{ijj}).$$

Hence

$$p^T(p\mathcal{T})p = \sum_{i=1}^{n} p_i \left\{\sigma_1(i) + p_i(3\sigma_2(i) + p_i \mathcal{T}_{iii})\right\}$$

requires $7n$ arithmetic operations plus the number of arithmetic operations to compute $\sigma_1(i)$ and $\sigma_2(i)$ for $i = 1, \ldots, n$. For $\sigma_1(i)$ the term $\sigma_3(i,j)$ must be computed for $f_i \leq j \leq i-1$ and requires $2(j - \max\{f_i, f_j\})$ number of arithmetic operations. Hence $\sigma_1(i)$ requires

$$5(i - f_i) + 1 + 2 \sum_{j=f_i}^{i-1} (j - \max\{f_i, f_j\}).$$

number of arithmetic operations. Computing $\sigma_2(i)$ requires $2(i - f_i)$ number of arithmetic operations. Hence, computing both $\sigma_1(i)$ and $\sigma_2(i)$ for all $i = 1, \ldots, n$ will require

$$\begin{aligned} &5 \sum_{i=1}^{n} (i - f_i) + n + 2 \sum_{i=1}^{n} \sum_{j=f_i}^{i-1} (j - \max\{f_i, f_j\}) + 2 \sum_{i=1}^{n} (i - f_i) \\ &= 2nnz(\mathcal{T}) + 5nnz(H) - 5n \end{aligned} \tag{27}$$

number of arithmetic operations. It follows that the number of arithmetic operations for computing $p^T(p\mathcal{T})p$ is $2\text{nnz}(\mathcal{T}) + 5\text{nnz}(H) + 2n$.

For $g = (p\mathcal{T})p \in \mathbb{R}^n$ the elements $1 \leq i \leq n$ are

$$g_i = \sum_{j=1}^{n} \sum_{k=1}^{n} p_j p_k \mathcal{T}_{ijk} \tag{28}$$

and the term $H = (p\mathcal{T}) \in \mathbb{R}^{n \times n}$ is given by

$$H_{ij} = \sum_{k=1}^{n} p_k \mathcal{T}_{ijk}, \;\; 1 \leq i, j \leq n. \tag{29}$$

In the next section we will give an algorithm for computing (28) and (29).

4.1 *Skyline implementation of:* $(p\mathcal{T})p$ *and* $(p\mathcal{T})$

The algorithm for the operation $(p\mathcal{T})p$ is shown in Algorithm *1* while the algorithm for the operation $(p\mathcal{T})$ is shown in Algorithm *2*.

Algorithm *1* Computes $g \leftarrow g + (p\mathcal{T})p$ when the tensor is induced by a structure of the skyline matrix.

Let $\mathcal{T} \in \mathbb{R}^{n\times n\times n}$ be a super-symmetric tensor.
Let $\{f_1, \ldots, f_n\}$ be the indices of the first nonzero elements for each row i of the matrix that induces the structure of the tensor $\mathcal{T}$.
Let $p, g \in \mathbb{R}^n$.
for $i = 1$ to n **do**
  for $j = f_i$ to $i - 1$ **do**
    $s = 0$
    for $k = \max\{f_i, f_j\}$ to $j - 1$ **do**
      $s+ = p_k \mathcal{T}_{ijk}$
      $g_k+ = 2p_i p_j \mathcal{T}_{ijk}$
    end for
    $g_i+ = p_j(2s + p_j \mathcal{T}_{ijj})$
    $g_j+ = 2p_i(s + p_j \mathcal{T}_{ijj})$
  end for
  $s = 0$
  for $k = f_i$ to $i - 1$ **do**
    $s+ = p_k \mathcal{T}_{iik}$
    $g_k+ = p_i^2 \mathcal{T}_{iik}$
  end for
  $g_i+ = p_i(2s + p_i \mathcal{T}_{iii})$
end for

To compute the number of arithmetic operations for Algorithm *1* and *2* we need first to compute the number of times each loop body is executed and then the number of arithmetic operations in each loop body.

The innermost **for** loop is executed $(\text{nnz}(\mathcal{T}) - 2\text{nnz}(H) + n)$ times. The **for** loop (i, j, j) is executed $(\text{nnz}(H) - n)$ times. The **for** loop (i, i, k) is executed $(\text{nnz}(H) - n)$ times. Finally the outermost **for** loop is executed n times.

Algorithm *2* Computes $H \leftarrow H + (p\mathcal{T})$ when the tensor is induced by a structure of the skyline matrix.

Let $\mathcal{T} \in \mathbb{R}^{n\times n\times n}$ be a super-symmetric tensor, then
$\{f_1, \ldots, f_n\}$ is the indices of the first nonzero elements for each row in H.
Let $p \in \mathbb{R}^n$ be a vector.
for $i = 1$ to n **do**
  for $j = f_i$ to $i - 1$ **do**
    for $k = \max\{f_i, f_j\}$ to $j - 1$ **do**
      $H_{ij} += p_k\mathcal{T}_{ijk}$
      $H_{ik} += p_j\mathcal{T}_{ijk}$
      $H_{jk} += p_i\mathcal{T}_{ijk}$
    end for
    $H_{ij} += p_j\mathcal{T}_{ijj}$
    $H_{jj} += p_i\mathcal{T}_{ijj}$
  end for
  for $k = f_i$ to $i - 1$ **do**
    $H_{ii} += p_k\mathcal{T}_{iik}$
    $H_{ik} += p_i\mathcal{T}_{iik}$
  end for
  $H_{ii} += p_i\mathcal{T}_{iii}$
end for

The number of arithmetic operations in Algorithm *2* is then $6(\text{nnz}(\mathcal{T}) - 2\text{nnz}(H) + n) + 4(\text{nnz}(H) - n) + 4(\text{nnz}(H) - n) + 2n$. The total number of arithmetic operations in Algorithm *2* is then $6\text{nnz}(\mathcal{T}) - 4\text{nnz}(H)$.

Note that we have implemented some code enhancements which are not shown in Algorithm *1*. We note that if $2p_ip_j$ is computed outside the innermost loop, then the body of this loop will require 4 arithmetic operations. The number of arithmetic operations in Algorithm *1* is then $4(\text{nnz}(\mathcal{T}) - 2\text{nnz}(H) + n) + 9(\text{nnz}(H) - n) + 4(\text{nnz}(H) - n) + 6n$. The total number of arithmetic operations in Algorithm *1* is then $4\text{nnz}(\mathcal{T}) + 5\text{nnz}(H) - 3n$.

A more detailed derivation of these two algorithms only using the super symmetry of the tensor is given in [15] showing code enhancements.

4.2 *LDL^T Factorization*

Algorithm *3* shows an LDL^T factorization of a sparse symmetric matrix [11]. The factorization is well defined if the matrix A is positive definite. We also outline the solve phase for a skyline structure.

Algorithm *3* Computing LDL^T where A is overwritten with the factors L and D.

Let $A \in \mathbb{R}^{n\times n}$ be a positive definite symmetric matrix.
Let $\{f_1, \ldots, f_n\}$ be the indices of the first nonzero elements for each row in A.
$a \in \mathbb{R}^n$ is a temporary vector.
for $k = 1$ to n **do**
  for $i = f_k$ to $k-1$ **do**
    $a_i = A_{ki}$
  end for
  for $j = f_k$ to $k-1$ **do**
    for $i = \max\{f_j, f_k\}$ to $j-1$ **do**
      $a_j = a_j - a_i A_{ji}$
    end for
    $A_{kj} = a_j / Ajj$
  end for
  for $i = f_k$ to $k-1$ **do**
    $A_{kk} = A_{kk} - a_i A_{ki}$
  end for
end for
{The matrix A contains now both L and D}
$\{L_{ij} \leftarrow A_{ij},\ i > j,\ L_{ii} = 1\}$
$\{D_{ii} \leftarrow A_{ii}\}$

We have as input in Algorithm *3* the $n \times n$ skyline symmetric positive definite matrix A. The output is an LDL^T factorization of A, where A is overwritten with the factors L and D. Since A is a positive definite symmetric matrix with skyline structure there is no fill-ins performing the LDL^T factorization and the structure is unchanged [11]. The Doolittle algorithm [11, see Chapter 10.2] for the LDL^T factorization has a very similar structure to that of the tensor operations $(p\mathcal{T})p$ and $(p\mathcal{T})$, shown in Algorithm *1* and *2*.

By using the same arguments as for Algorithm *1* and *2*, it can be shown that Algorithm *3* requires $2\text{nnz}(\mathcal{T}) - \text{nnz}(H) - n$ number of arithmetic operations.

The solve phase $Ax = b$, $A = LDL^T$ involves solving the sets of equations

$$Ly = b\ ,\ Dz = y \text{ and } L^T x = z.$$

Algorithm 4 Solves $LDL^T x = b$.

Let $L \in \mathbb{R}^{n \times n}$ be a lower triangular matrix.
Let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix.
Let $b \in \mathbb{R}^n$.
for $k = 2$ to n **do**
  for $j = f_k$ to $k - 1$ **do**
    $b_k = b_k - L_{kj} b_j$
  end for
end for
for $k = 1$ to n **do**
  $b_k = b_k / D_{kk}$
end for
for $k = n$ to 1 **do**
  for $j = f_k$ to $k - 1$ **do**
    $b_j = b_j - L_{kj} b_k$
  end for
end for

Algorithm 4 gives the solve phase of $LDL^T x = b$. The temporary vectors y, and z and x can be overwritten by the initial vector b, through the solve phase.

Algorithm 4 requires $4\text{nnz}(H) - 3n$ number of arithmetic operations.

The algorithm for the LDL^T factorization is row-oriented, thus the data structure layout is done row-wise. We have used a linear array to store both the Hessian matrix and its induced tensor. However jagged arrays can perform just as well as linear arrays considering efficiency [16]. The jagged variable band storage discussed in [17] can be used for a symmetric skyline structure.

In the next section the algorithms for performing Newton's method and the methods of the Halley class are shown and the costs for both operations and the total methods are summarized.

5 The Halley Class and Unconstrained Optimization

In section 2 we gave the derivation of the Halley class (3) for solving systems of nonlinear equations as two iterations of Newton's method applied to the second order Taylor approximation of the function. Consider the unconstrained

optimization problem (2). Define

$$g^{(k)} = \nabla f(x_k), H^{(k)} = \nabla^2 f(x_k), \text{ and } \mathcal{T}^{(k)} = \nabla^3 f(x_k). \tag{30}$$

Recall the notation $(p\mathcal{T}^{(k)})$ is the matrix $\nabla^3 f(x_k)p$. Also note that when we multiply the tensor $\mathcal{T}$ with a vector $p \in \mathbb{R}^n$ the product is $(p\mathcal{T})$ is a symmetric matrix with the same sparsity structure as the matrix inducing the tensor $\mathcal{T}$. To see this consider (29) and pair of indices (i, j) so that $H_{ij} = 0$. Then, from (16), $\mathcal{T}_{ijk} = 0$ for all k and $(p\mathcal{T})_{ij} = 0$.

One iteration of a method in the Halley class for $F(x) \equiv \nabla f(x) = 0$ is

$$\begin{aligned} H^{(k)} s_k^{(1)} &= -g^{(k)} \\ \left(H^{(k)} + \alpha(s_k^{(1)}\mathcal{T}^{(k)})\right) s_k^{(2)} &= -\frac{1}{2}(s_k^{(1)}\mathcal{T}^{(k)})s_k^{(1)} \\ x_{k+1} &= x_k + s_k^{(1)} + s_k^{(2)}. \end{aligned}$$

To simplify the notation, we eliminate the iteration index in the algorithm. One iteration of a method in the Halley class is divided into 13 minor steps.

0) Evaluate $g = \nabla f(x_k)$, $H = \nabla^2 f(x_k)$, and $\mathcal{T} = \nabla^3 f(x_k)$.
1) Factorize the matrix H: $LDL^T = H$.
2) Solve for $s^{(1)}$: $LDL^T s^{(1)} = g$.
3) Set $s = s^{(1)}$.
4) Compute $\widehat{s}$: $\widehat{s} = \alpha s$.
5) Compute $\widehat{H}$: $\widehat{H} = H - (\widehat{s}\mathcal{T})$.
6) Compute the matrix-vector product: $y = \widehat{H}s$.
7) Compute the difference $g - y$.
8) Compute $\widehat{g}$: $\widehat{g} = (\frac{1}{2\alpha})(g - y)$.
9) Factorize the matrix $\widehat{H}$: $LDL^T = \widehat{H}$.
10) Solve for $s^{(2)}$: $LDL^T s^{(2)} = \widehat{g}$.
11) Update the step $s = s^{(1)} + s^{(2)}$.
12) Update iterate $x_{k+1} = x_k - s$.

For Newton's method the steps 4 to 11 are omitted. For Super Halley step 4 can be simplified since $\alpha = 1$. Step 5 is computed with Algorithm *2*. For the Chebyshev's method the steps 4 to 9 are combined and replaced by the step compute $\widehat{g} = \frac{1}{2}(s\mathcal{T})s$ computed with Algorithm *1*. In the steps 7 and 8 we observe that $y = \widehat{H}s = Hs - (\widehat{s}\mathcal{T})s = g - (\widehat{s}\mathcal{T})s$ using s is the solution of $Hs = g$ in step 2 so $(\widehat{s}\mathcal{T})s = g - y$. Hence $\widehat{g} = \frac{1}{2}(s\mathcal{T})s$.

The LDL^T decomposition in the steps 1 and 10 are computed with Algorithm *3* and the steps 2 and 11 use Algorithm *4*. The symmetric matrix

vector operation in step 6 requires $4\text{nnz}(H) - 3n$ arithmetic operations. Step 3 involves no arithmetic operations.

The computational requirements in terms of floating point arithmetic operations for each step in the algorithm outlined above is shown in Table 1. The columns marked dense, banded and skyline refers to the matrix structure and the tensor induced by the structure of the matrix.

Table 1. Number of floating point arithmetic cost for each step

Step	Dense	Banded	Skyline
1, 9	$\frac{1}{3}n^3 + \frac{1}{2}n^2 - \frac{5}{6}n$	$n\beta^2 + 2n\beta - \frac{2}{3}\beta^3 - \frac{3}{2}\beta^2 - \frac{5}{6}\beta$	$2\text{nnz}(\mathcal{T}) - \text{nnz}(H) - n$
2, 10	$2n^2 - n$	$4n\beta + n - 2\beta^2 - 2\beta$	$4\text{nnz}(H) - 3n$
5	$n^3 + n^2$	$3n\beta^2 + 5n\beta + 2n - 2\beta^3 - 4\beta^2 - 2\beta$	$6\text{nnz}(\mathcal{T}) - 4\text{nnz}(H)$
(4-9)*	$\frac{2}{3}n^3 + \frac{9}{2}n^2 + \frac{5}{6}n$	$2n\beta^2 - \frac{4}{3}\beta^3 + 11n\beta - \frac{3}{2}\beta^2 + 6n - \frac{31}{6}\beta$	$4\text{nnz}(\mathcal{T}) + 5\text{nnz}(H) - 3n$
6	$2n^2 - n$	$4n\beta + n - 2\beta^2 - 2\beta$	$4\text{nnz}(H) - 3n$
4, 7, 8, 11, 12	n	n	n

The row marked (4-9)* is for Chebyshev's method where the steps 4 to 9 are replaced by a single step $\widehat{g} = \frac{1}{2}(s\mathcal{T})s$ computed by Algorithm *1*.

The total computational requirements for each method is shown in Table 2. The computational requirements show that the Halley class and Newton's method have the same asymptotic upper bound for the dense, banded and skyline structure.

Table 2. Number of arithmetic operations for one iteration.

Method	Dense	Banded	Skyline
Newton	$\frac{1}{3}n^3 + \frac{5}{2}n^2 - \frac{5}{6}n$	$n\beta^2 - \frac{2}{3}\beta^3 + 6n\beta - \frac{7}{2}\beta^2 + 2n - \frac{17}{6}\beta$	$2\text{nnz}(\mathcal{T}) + 3\text{nnz}(H) - 3n$
Chebyshev	$n^3 + 9n^2 + n$	$3n\beta^2 - 2\beta^3 + 21n\beta - 18\beta^2 + 11n - 10\beta$	$6\text{nnz}(\mathcal{T}) + 12\text{nnz}(H) - 7n$
Halley	$\frac{5}{3}n^3 + 5n^2 + \frac{4}{3}n$	$5n\beta^2 - \frac{10}{3}\beta^3 + 21n\beta - 16\beta^2 + 10n - \frac{29}{3}\beta$	$10\text{nnz}(\mathcal{T}) + 6\text{nnz}(H) - 6n$
Super Halley	$\frac{5}{3}n^3 + 5n^2 + \frac{1}{3}n$	$5n\beta^2 - \frac{10}{3}\beta^3 + 21n\beta - 16\beta^2 + 9n - \frac{28}{3}\beta$	$10\text{nnz}(\mathcal{T}) + 6\text{nnz}(H) - 7n$

The storage requirements are summarized in Table 3

Table 3. Memory usage for the Newton and Halley class methods.

Method	Dense	Banded	Skyline
Newton	$\frac{1}{2}n^2 + \frac{7}{2}n$	$n\beta + 4n$	$\text{nnz}(H) + 3n$
Chebyshev	$\frac{1}{6}n^3 + n^2 + \frac{17}{3}n$	$\frac{1}{2}n\beta^2 + \frac{5}{2}n\beta + 7n$	$\text{nnz}(\mathcal{T}) + \text{nnz}(H) + 5n$
Halley Class	$\frac{1}{6}n^3 + 2n^2 + \frac{17}{3}n$	$\frac{1}{2}n\beta^2 + \frac{9}{2}n\beta + 9n$	$\text{nnz}(\mathcal{T}) + 2\text{nnz}(H) + 5n$

5.1 *Halley's method versus Newton's method*

Given the computational requirements in Table 1 we can now find the number of floating point arithmetic operations for one iteration of a method in the Halley class and one iteration of Newton's method excluding the cost of

evaluating the derivatives (30) (in step 0) provided the Hessian matrix has a skyline structure.

Let W_{Halley} be the number of arithmetic operations for one iteration of a method in the Halley class. Then W_{Halley} is the sum of arithmetic operations in each of the steps 1 through 12. Let W_{Newton} be the number of arithmetic operations for one iteration of Newton's method. Then W_{Newton} is the sum of arithmetic operations in each of the steps 1 through 3 and step 12.

THEOREM 5.1 *Assume that matrices in steps 2 and 10 are positive definite and has a skyline sparsity structure. Then*

$$\frac{W_{Halley}}{W_{Newton}} \leq 5.$$

Proof The number of arithmetic operations for one iteration of the methods in the Halley class and one iteration of Newton's method are given in Table 2. Then

$$\begin{aligned} W_{\text{Newton}} &= 2\text{nnz}(\mathcal{T}) + 3\text{nnz}(H) - 3n \text{ and} \\ W_{\text{Halley}} &= 10\text{nnz}(\mathcal{T}) + 6\text{nnz}(H) - 6n \end{aligned}$$

and we have

$$\frac{W_{\text{Halley}}}{W_{\text{Newton}}} = 5 - \frac{9(\text{nnz}(H) - n)}{2nnz(\mathcal{T}) + 3\text{nnz}(H) - 3n} \leq 5$$

using that $\text{nnz}(H) \geq n$. Since the matrices are positive definite the diagonal elements are nonzero. □

This result shows that the ratio of the number of arithmetic operations of one iteration a method in the Halley class and Newton's method is bounded by a constant independent of the size n and the sparsity structure of the Hessian matrix as long as it is a skyline structure. The skyline structure includes dense and banded matrices. This upper bound is demonstrated by numerical results.

5.1.1 ***Actual Sparsity of a Tensor.*** The actual number of nonzero elements in the tensor can be much smaller than the number of elements in the induced tensor. Let $F : \mathbb{R}^n \to \mathbb{R}^m$ where component ℓ of F at x is $F_\ell(x)$ and consider the least squares formulation

$$\min_{x \in \mathbb{R}^n} f(x) \equiv \frac{1}{2}||F(x)||_2^2 = \frac{1}{2}\sum_{\ell=1}^{m} F_\ell(x)^2.$$

Assume that each component $F_\ell(x)$ is separable and only depends on a few variables with indices in $\chi_\ell \subseteq \{1, 2, ..., n\}$ then

$$F_\ell(x) = \sum_{i \in \chi_\ell} \varphi_{\ell i}(x_i), \quad \ell = 1, 2, ..., m$$

where x_i is component i of the vector x. Further let J_i be the set of indices of the element functions $F_\ell(x)$ that depends on variable x_i. Then

$$J_i = \{\ell : i \in \chi_\ell\}$$

and element ij of the Hessian matrix at x is

$$\frac{\partial^2}{\partial x_i \partial x_j} f(x) = \sum_{\ell \in J_i \cap J_j} \varphi'_{\ell i}(x_i) \varphi'_{\ell j}(x_j)$$

and it follows that

$$\mathcal{T}_{ijk} = 0 \text{ when } i \neq j, i \neq k, \text{ and } j \neq k.$$

If this is utilized in the Algorithms *1* and *2* the number of floating point operations will be reduced. The innermost **for** loop is not traversed in any of the Algorithms *1* and *2* when $\mathcal{T}_{ijk} = 0$ for $i \neq j, i \neq k,$ and $j \neq k$. The number of arithmetic operations for the algorithms utilizing actual sparsity is shown in Table 4.

Table 4. The number of arithmetic operations for induced and actual sparsity tensor operations.

Operations	Induced	Actual
$(p\mathcal{T})$	$6\text{nnz}(\mathcal{T}) - 4\text{nnz}(H)$	$8\text{nnz}(H) - 6n$
$(p\mathcal{T})p$	$4\text{nnz}(\mathcal{T}) + 5\text{nnz}(H) - 3n$	$9\text{nnz}(H) - 3n$

Thus $W_{\text{Halley}} = 4\text{nnz}(\mathcal{T}) + 18\text{nnz}(H) - 12n$ while the number of arithmetic operations for one iteration of Newton's method will not change $W_{\text{Newton}} = 2\text{nnz}(\mathcal{T}) + 3\text{nnz}(H) - 3n$ as shown in Table 2. Thus we get the upper bound for the ratio of arithmetic operations of one iteration of a method in the Halley class to one iteration of Newton's method. The result in the theorem follows directly from the reduction in Algorithm *2* to $8\text{nnz}(H) - 6n$ arithmetic operations. Using the expressions for $\text{nnz}(\mathcal{T})$ in (24) and $\text{nnz}(H)$ in (23)

$$\frac{W_{\text{Halley}}}{W_{\text{Newton}}} = \frac{2n\beta^2 - \frac{4}{3}\beta^3 + 24n\beta - 13\beta^2 + 10n - \frac{35}{3}\beta}{n\beta^2 - \frac{2}{3}\beta^3 + 6n\beta - \frac{7}{2}\beta^2 + 2n - \frac{17}{6}\beta}.$$

Similarly, using the expressions for $\mathrm{nnz}(\mathcal{T})$ in (22) and $\mathrm{nnz}(H)$ in (21)

$$\frac{W_{\text{Halley}}}{W_{\text{Newton}}} = \frac{\frac{2}{3}n^3 + 11n^2 - \frac{5}{3}n}{\frac{1}{3}n^3 + \frac{5}{2}n^2 - \frac{5}{6}n}.$$

We have not used that Super Halley ($\alpha = 1$) and Chebyshev ($\alpha = 0$) can be computed with fewer arithmetic operations.

THEOREM 5.2 *If the matrices are positive definite with a band structure with half bandwidth β and that $\mathcal{T}_{ijk} = 0$ where $i \neq j \neq k \neq i$ is utilized in Algorithm 2 then*

$$\frac{W_{Halley}}{W_{Newton}} = \frac{2\beta^2 + 24\beta + 10}{\beta^2 + 6\beta + 2} + O(\frac{1}{n}).$$

If the matrix is dense then $\frac{W_{Halley}}{W_{Newton}} = 2 + O(\frac{1}{n})$.

We can now compare the use of Halley's and Newton's methods for a band matrix for increasing bandwidth β. In Table 5 we see as the bandwidth increases that the number of arithmetic operations for one iteration of Halley's methods is close to two times the number for Newton's method when the actual sparsity is utilized

Table 5. $W_{\text{Halley}}/W_{\text{Newton}}$ as a function of β using actual and induced sparsity.

β	Actual	Induced
0	5	5
1	$4 + O(\frac{1}{n})$	$4 + O(\frac{1}{n})$
2	$3\frac{2}{3} + O(\frac{1}{n})$	$4 + O(\frac{1}{n})$
$n-1$	$2 + O(\frac{1}{n})$	$5 - O(\frac{1}{n})$

The last line in Table 5 follows from the dense case. It should be noted that for $\beta \leq 1$ there is no difference between actual and induced sparsity. The storage requirement for Halley's method is also reduced to n^2 in the dense case, $2n\beta + n$ in the banded case, and $2\mathrm{nnz}(H) - n$ in the general case for a matrix with skyline structure.

6 Numerical Results

We consider three test problems where the Hessian matrix has a skyline structure. Chained Rosenbrock [18]:

$$f(x) = \sum_{i=2}^{n} [6.4(x_{i-1} - x_i^2)^2 + (1 - x_i)^2].$$

Generalized Rosenbrock [19]:

$$f(x) = \sum_{i=1}^{n-1} [(x_n - x_i^2)^2 + (x_i - 1)^2].$$

Broyden Banded [20]:

$$f(x) = \sum_{i=1}^{n} [x_i(2 + 15x_i^2) + 1 - \sum_{j \in J_i} x_j(1 + x_j)]^2$$

where

$$J_i = \{j : j \neq i, \ \max\{1, i - m_l\} \leq j \leq \min\{n, i + m_u\}\}$$

and $m_l = 5$ and $m_u = 1$.

The Chained Rosenbrock function has a tri-diagonal Hessian, see Figure 2, for all x. The Generalized Rosenbrock function has an arrowhead Hessian, see Figure 1, for all x. The Broyden Banded function has a banded Hessian, with bandwidth $\beta = 6$, for all x.

We see that both Chained Rosenbrock and Broyden Banded are k-chained functions where f_i is monotonically increasing as described in Theorem 3.2.

6.1 *Chained Rosenbrock*

In Figure 5 we have the elapsed time ratio with function, gradient, Hessian and tensor evaluations (a), the elapsed time ratio without any function, gradient, Hessian and tensor evaluations (b), and when we utilize actual sparsity for one step Halley and one step Newton with function, gradient, Hessian and tensor evaluations (c). Finally the theoretical bound for the Halley and Newton's method (d). For Chained Rosenbrock $\mathcal{T}_{ijk} = 0$ where $i \neq j \neq k \neq i$. Utilizing this actual sparsity the time savings were on average 10%.

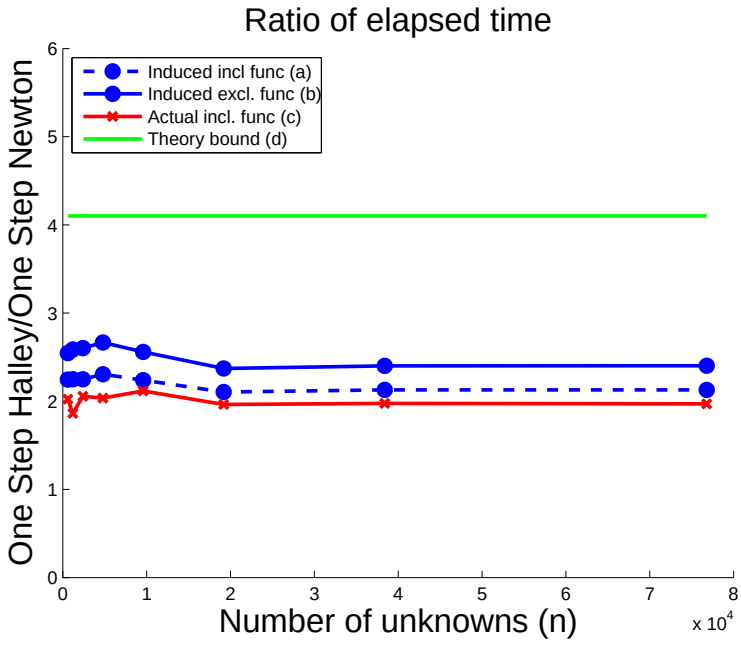

Figure 5. Chained Rosenbrock.

Figure 5 shows that the ratio is constant for increasing number of unknowns (n) and is bounded by the theoretical bound. The difference between the theoretical bound and fraction of elapsed time is partially due to the loop overhead in the Algorithm *2* and *3* when β is small.

The Hessian of the Chained Rosenbrock function is a tri-diagonal matrix with $\beta = 1$. There is no difference in number of arithmetic operations between induced and actual sparsity for $\beta = 1$. The minor difference in Figure 5 is due to the induced method which uses a general code while for actual sparsity the **for** loops which are not traversed are removed.

6.2 *Generalized Rosenbrock*

In Figure 6 we have the elapsed time ratio with function, gradient, Hessian and tensor evaluations (a), the elapsed time ratio without any function, gradient, Hessian and tensor evaluations (b), and when we utilize actual sparsity for one step Halley and one step Newton (c). For Generalized Rosenbrock $\mathcal{T}_{ijk} = 0$ where $i \neq j \neq k \neq i$, when utilizing this actual sparsity the time savings were on average 10%.

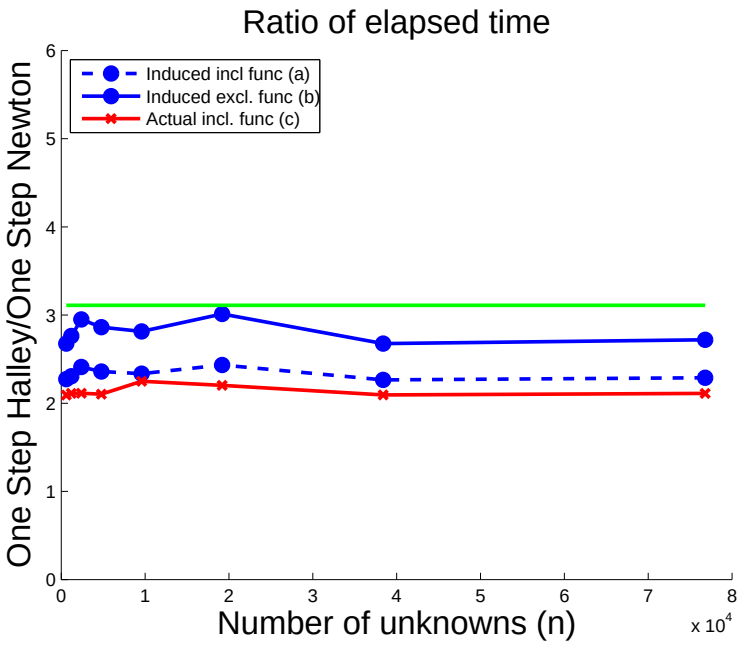

Figure 6. Generalized Rosenbrock.

Figure 6 shows that the ratio is constant, and bounded by 3, for increasing number of unknowns (n) and is close to the theoretical bound.

6.3 *Broyden Banded*

In Figure 7 we have the elapsed time ratio without any function, gradient, Hessian and tensor evaluations (a), the elapsed time ratio with function, gradient, Hessian and tensor evaluations (b) and when we utilize actual sparsity for one step Halley and one step Newton with function, gradient, Hessian and tensor evaluations (c). Finally the theoretical bound for the Halley and Newton's method (d). For Broyden Banded $\mathcal{T}_{ijk} = 0$ where $i \neq j \neq k \neq i$, when utilizing this actual sparsity the time savings were on average 10%.

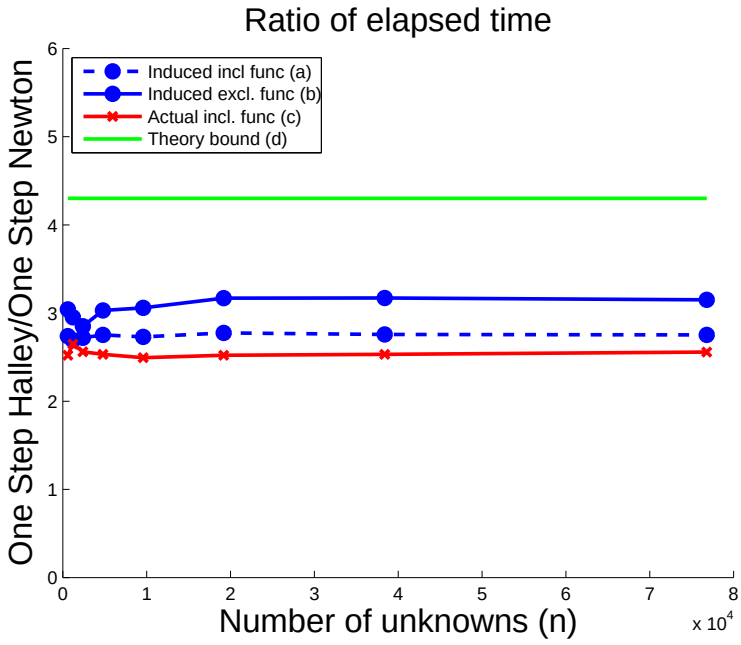

Figure 7. Broyden Banded.

Figure 7 shows that the ratio is constant for increasing number of unknowns (n) and is close to the theoretical bound.

The difference in the theoretical bound and the fraction of elapsed time is partially due to the loop overhead in the Algorithm *2* and *3* when β is small. For Broyden Banded we have the possibility to increase the bandwidth of the Hessian, β, and for β large the elapsed time ratio without any function, gradient, Hessian and tensor evaluations is over 4.

In Figure 8 we compare Newton's method and Super Halley's method on the Broyden Banded test function. This figure shows that Newton's method has more iterations than Super Halley's method to reach the same accuracy of the norm of the gradient, $||\nabla f||$. For both Super Halley and Newton's method the elapsed time includes function, gradient, Hessian and tensor evaluations.

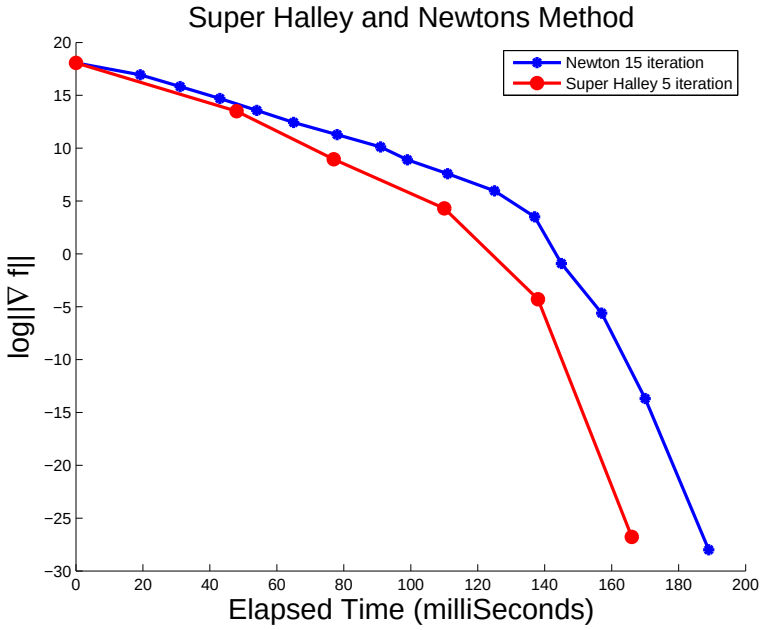

Figure 8. Banded Problems

7 Conclusions

We have shown that for a large class of sparse matrices the ratio of the number of arithmetic operations of Halley's and Newton's method is constant per iteration independent of the number of unknowns (n).

Limited numerical testing using chained partially separable test functions indicate that the elapsed time ratio of Halley's and Newton's method including the time to compute the gradient and the relevant higher derivatives has the same favorable upper bound.

We have shown that the number of nonzero elements in the tensor can be much smaller than the number of nonzero elements indicated by the structure of the induced tensor. Utilizing this reduces the ratio of elapsed time of Halley's and Newton's method.

Acknowledgement

We want to thank the anonymous referee for careful evaluation of this paper and Hubert Schwetlick for pointing out the early history of the Halley class.

References

[1] Yamamoto, T., 1988, On the method of tangent hyperbolas in Banach spaces. *Journal of Computational and Applied Mathematics*, **21**, 75–86.

[2] Yamamoto, T., 2000, Historical developments in convergence analysis for Newton's and Newton-like methods. *Journal of Computational and Applied Mathematics*, **124**, 1–23.

[3] Werner W., 1980, Some improvement of classical iterative methods for the solution of nonlinear equations. *In: E.L. Allgower, K. Glashoff and N.-O. Peitgen, Eds., Proc. Numerical Solution of Nonlinear Equations (Springer, Bremen)*, 426–440.

[4] Gutiérrez, J. M. and Hernández, M. A., 2001, An acceleration of Newton's method: Super-Halley method. *Applied Mathematics and Computation*, **117**(2), 223–239.

[5] Nečepuerenko, M. I., 1954, On Čebyšev's method for functional equations. *Uspehi Matem. Nauk(NS)*, **9**(2), 163–170.

[6] Halley, E., 1694, A new, exact, and easy method of finding the roots of any equations generally, and that without any previous reduction. *Philos. Trans. Roy. Soc. London*, **18**, 136–145.

[7] Mertvecova, M. A., 1953, Analogue of the process of tangent hyperbolas for general functional equations. *Doklady Akad. Nauk SSSR(NS)*, **88**, 611–614.

[8] Chen, D., Argyros, I. K., and Qian, Q., 1994, A Local Convergence Theorem for the Super-Halley Method in a Banach Space. *Appl. Math. Lett*, **7**(5), 49–52.

[9] Schwetlick, H., 1979, Numerische Lösung nichtlinearer Gleichungen. *R. Oldenbourg Verlag, München-Wien*.

[10] Griewank, A. and Toint, Ph. L., 1982, *On the unconstrained optimization of partially separable functions*. In: Nonlinear Optimization, M. J. D. Powell (Ed.), Academic Press, New York, NY, pp. 301–312.

[11] Duff, I.S., Erisman, A.M., and Reid, J.K., 1986, *Direct Methods for Sparse Matrices*, Oxford University Press.

[12] Knuth, D.E., 1997, *The Art of Computer Programming, Volume 1 (3rd ed.): Fundamental Algorithms*, Addison-Wesley.

[13] Bagirov, A.M., and Ugon, J., 2006, Piecewise Partially Separable Functions and a Derivative-free Algorithm for Large Scale Nonsmooth Optimization. *Journal of Global Optimization*, **35**, 163–195.

[14] Conn, A.R., Gould, N., and Toint., Ph.L., 1988, Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, vol, **50**(182), 399–430.

[15] Gundersen, G., and Steihaug, T., 2006, On computing with general sparse third derivatives in unconstrained optimization. I: *NIK'2006 : Norsk informatikkonferanse*, ISBN 82-519-2186-4., Trondheim: Tapir Akademisk Forlag, 53–64.

[16] Lujan, M., Usman, A., Hardie, P., Freeman, T. L., and Gurd, J. R., 2005, Storage Formats for Sparse Matrices in Java. *In proceedings of the 5th International Conference on Computational Science - ICCS 2005, Lecture Notes in Computer Science 3514.*, 364–371.

[17] Gundersen, G., and Steihaug, T., 2004, On The Use of Java Arrays for Sparse Matrix Computations. *Parallel Computing: Software Technology, Algorithms, Architectures and Applications*, **13**, North Holland/Elsevier, 119–126.

[18] Toint, Ph. L., 1978, Some numerical results using a sparse matrix updating formula in unconstrained optimization. *Mathematics of Computation*, **32**(143), 839–851.

[19] Schwefel, H. P., 1981, *Numerical Optimization of Computer Models*, John Wiley and Sons, Chichester.

[20] Moré, J. J., Garbow, B. S., and Hillstrom, K. E., 1981, Testing Unconstrained Optimization Software. *ACM Transaction on Mathematical Software*, **7**, 17–24.