# Approximating Large-Scale Hessian Matrices Using Secant Equations

Jennifer Scott

STFC Rutherford Appleton Laboratory and
the University of Reading

This work is in collaboration with Jaroslav Fawkes and Nick Gould (RAL).
It was funded by an EPSRC Mathematics Small Grant.

Bath RAL Day, 10 April 2025

Consider the optimization problem

$$\min_x f(x),$$

where $f(x)$ is twice differentiable function of $n$ variables.

Gradient

$$g(x) := \nabla f(x)$$

Hessian matrix

$$H(x) := \nabla^2 f(x)$$

Our interest is $n$ large and $H(x)$ sparse

# Summary

- Large-scale optimization algorithms frequently require large sparse Hessian matrices $H(x)$
  For example, in a trust region method.
- How can we obtain good approximations to $H(x)$?
- We want fast and efficient approximation algorithm with scope to exploit parallelism.
- We propose a novel approach that seeks to satisfy as many componentwise secant equations as is necessary to define each row of the Hessian matrix.

# Why do we need an approximation?

In optimisation algorithms, there are both theoretical and practical benefits of having $H(x)$.

In practice, $H(x)$ is generally not readily available.

- Analytic expression unlikely.
- Backward mode of automatic differentiation, $g(x)$ costs small multiple of evaluating $f(x)$
- BUT $H(x)$ costs $\mathcal{O}(n)$ times that of $f(x)$.

**Key issue:** $n$ is LARGE

# Challenge

Build approximations $B^{(k)} = \{b_{ij}^{(k)}\}$ of $H(x)$ at a sequence of iterates $x^{(k)}$.

- Interest in approximating $H(x)$ dates back to 1960s.
- Focus was on $n$ small and $H(x)$ dense.
- Extensions to sparse case unsuccessful because either the formulae used generated dense matrices (impractical for large problems), or imposing sparsity led to numerical instability in the approximation algorithms.
- In 1990s, limited-memory strategy proposed (Fletcher, Grothey and Leyffer) that incorporated the curvature observed at a number of previous iterates. But no attempt to impose sparsity.

# Secant equations

Let $H^{(k)} = H(x^{(k)})$

Properties: $H^{(k)}$ is $n \times n$ symmetric matrix with known sparsity pattern.

**Aim:** use previously accumulated information to estimate $B^{(k)} \approx H^{(k)}$.

Define $s^{(l)} := x^{(l)} - x^{(l-1)}$ and $y^{(l)} := g(x^{(l)}) - g(x^{(l-1)})$

Construct $B^{(k)}$ that best satisfies the multiple *secant conditions*

$$B^{(k)} s^{(l)} = y^{(l)}, \qquad l = k - m + 1, \ldots, k$$

where $m$ is the number of past (known) data pairs $\{s^{(l)}, y^{(l)}\}$

# Secant equations

Construct $B^{(k)}$ that best satisfies the multiple *secant conditions*

$$B^{(k)}s^{(l)} = y^{(l)}, \qquad l = k - m + 1, \ldots, k.$$

For each $k$, solve convex quadratic programming problem

$$\min_{B^{(k)}} \sum_{l=k-m+1}^{k} \|B^{(k)}s^{(l)} - y^{(l)}\|_F^2$$

such that

$$B^{(k)} = (B^{(k)})^T \quad \text{and} \quad \mathcal{S}(B^{(k)}) = \mathcal{S}(H^{(k)}).$$

Here $\mathcal{S}(B) := \{(i,j) : b_{ij} \neq 0\}$ denotes sparsity pattern of $B$.

Prohibitively expensive for large $n$ within optimization code.

# New approach

Start with the case that all rows of $H^{(k)}$ are sparse.

Instead of imposing full secant conditions for each row $i$, we will satisfy as many *componentwise* equations

$$e_i^T B^{(k)} s^{(l)} = e_i^T y^{(l)}, \quad l = k, k-1, \ldots,$$

as are necessary to define the row.

Set $\mathcal{S}_i^{(k)} := \{j : h_{ij}^{(k)} \neq 0\}$ and $nz_i := |\mathcal{S}_i^{(k)}|$

Then for $l = k, k-1, \ldots, k - nz_i + 1$ we require

$$\sum_{j \in \mathcal{S}_i^{(k)}} b_{ij}^{(k)} s_j^{(l)} = y_i^{(l)}$$

Let $z_i^{(k)}$ denote the vector of entries in row $i$

Rewriting, $z_i^{(k)}$ is the solution of $nz_i \times nz_i$ dense linear system

$$A_i^{(k)} z_i^{(k)} = c_i^{(k)} \qquad (1)$$

where $(A_i^{(k)})_{lj} = s_j^{(l)}$ and $(c_i^{(k)})_l = y_i^{(l)}$.

**Example 1** Consider the $3 \times 3$ approximate Hessian matrix

$$B^{(k)} = \begin{pmatrix} b_{11}^{(k)} & b_{12}^{(k)} & 0 \\ b_{21}^{(k)} & 0 & b_{23}^{(k)} \\ 0 & b_{32}^{(k)} & b_{33}^{(k)} \end{pmatrix}, \quad \text{with } b_{12}^{(k)} = b_{21}^{(k)} \text{ and } b_{23}^{(k)} = b_{32}^{(k)}.$$

For row $i = 2$, $\mathcal{S}_2^{(k)} = \{1, 3\}$, $nz_2 = 2$ and system (1) is

$$\underbrace{\begin{pmatrix} s_1^{(k)} & s_3^{(k)} \\ s_1^{(k-1)} & s_3^{(k-1)} \end{pmatrix}}_{A_2^{(k)}} \underbrace{\begin{pmatrix} b_{21}^{(k)} \\ b_{23}^{(k)} \end{pmatrix}}_{z_2^{(k)}} = \underbrace{\begin{pmatrix} y_2^{(k)} \\ y_2^{(k-1)} \end{pmatrix}}_{c_2^{(k)}}.$$

## Algorithm 1: Sparse Hessian approximation (row-wise independent)

1: **for** $i = 1, \ldots, n$ **do**
2: Compute row $i$ by constructing and solving system (1).
3: **end for**
4: Symmetrise $B^{(k)} := (B^{(k)} + (B^{(k)})^T)/2$.

Important: the rows can be computed in parallel in any order.

But averaging the off-diagonal entries does not truly take symmetry into account.

Also, only one triangle of $B^{(k)}$ may be available.

Rewrite the system

$$A_i^{(k)} z_i^{(k)} = c_i^{(k)} \qquad (1)$$

$$U_i^{(k)} \tilde{z}_i^{(k)} = c_i^{(k)} - K_i^{(k)} w_i^{(k)}, \qquad (2)$$

where

- $w_i^{(k)}$ holds the entries in row $i$ of $B^{(k)}$ that are already known.
- $\tilde{z}_i^{(k)}$ holds the $nu_i$ unknown entries in row $i$.
- $U_i^{(k)}$ is a square matrix of order $nu_i \leq nz_i$.

**Example 2** Consider the $5 \times 5$ approximate Hessian matrix

$$B^{(k)} = \begin{pmatrix} b_{11}^{(k)} & 0 & 0 & b_{14}^{(k)} & 0 \\ 0 & b_{22}^{(k)} & 0 & b_{24}^{(k)} & 0 \\ 0 & 0 & 0 & b_{34}^{(k)} & 0 \\ b_{41}^{(k)} & b_{42}^{(k)} & b_{43}^{(k)} & b_{44}^{(k)} & b_{45}^{(k)} \\ 0 & 0 & 0 & b_{54}^{(k)} & b_{55}^{(k)} \end{pmatrix}.$$

Assume rows are computed in the natural order $1, 2, 3, 4, 5$.

In row 4, $b_{41}^{(k)}, b_{42}^{(k)}, b_{43}^{(k)}$ are already known by symmetry. Hence $nu_4 = 2$.

$$\underbrace{\begin{pmatrix} s_4^{(k)} & s_5^{(k)} \\ s_4^{(k-1)} & s_5^{(k-1)} \end{pmatrix}}_{U_4^{(k)}} \underbrace{\begin{pmatrix} b_{44}^{(k)} \\ b_{45}^{(k)} \end{pmatrix}}_{\tilde{z}_4^{(k)}} = \underbrace{\begin{pmatrix} y_4^{(k)} \\ y_4^{(k-1)} \end{pmatrix}}_{c_4^{(k)}} - \underbrace{\begin{pmatrix} s_1^{(k)} & s_2^{(k)} & s_3^{(k)} \\ s_1^{(k-1)} & s_2^{(k-1)} & s_3^{(k-1)} \end{pmatrix}}_{K_4^{(k)}} \underbrace{\begin{pmatrix} b_{14}^{(k)} \\ b_{24}^{(k)} \\ b_{34}^{(k)} \end{pmatrix}}_{w_4^{(k)}}.$$

The order in which the rows are processed is now important.

## Algorithm 2: Sparse Hessian approximation (row-wise dependent)

1: Compute adjacency graph $\mathcal{G}(B^{(k)})$ and degree* of each vertex.
2: **for** $i = 1, \ldots, n$ **do**
3: Select vertex $v$ of minimum degree; let corresponding row be $row_i$.
4: Compute the $nu_i$ unknown entries in $row_i$ by solving a linear system of order $nu_i$.
5: Remove $v$; decrement degree of each remaining neighbour by 1.
6: **end for**

* initially the degree is the number of entries in the corresponding row

Problems:

- serial approach;
- inaccurate estimates from earlier steps can be magnified when solving for current row, leading to large error growth, particularly if some rows have a large number of entries. This happens in practice!

## Numerical experiments to illustrate the issues

- Examples from CUTEst test set
- Results for fixed Hessian $H(x^{(k)}) = H$ for all $k$.
- Randomly generate $s^{(l)} \in (-1, 1)$ and then compute $y^{(l)} = Hs^{(l)}$ for $l = 1, \ldots, m$.
- If the linear system to be solved for the unknowns in a row is under or over determined then compute the linear least squares solution (using an SVD decomposition).

Check results using maximum and median relative componentwise error

$$max\_rel\_err = \max_{(i,\, j)\in\mathcal{S}(H)} |b_{ij} - h_{ij}|/\max(1, |h_{ij}|),$$

$$med\_rel\_err = \operatorname{median}_{(i,\, j)\in\mathcal{S}(H)} |b_{ij} - h_{ij}|/\max(1, |h_{ij}|),$$

## Results with $m = 100$

| identifier | $nnz(row)$ | Algorithm 1 | | Algorithm 2 | |
|---|---|---|---|---|---|
| | | *max_rel_err* | *med_rel_err* | *max_rel_err* | *med_rel_err* |
| CURLY30 | 61 | 8.83E-10 | 9.69E-15 | 6.00E-06 | 9.29E-12 |
| NCVXBQP1 | 9 | 1.41E-09 | 1.22E-15 | ‡ | 2.59E+01 |
| SINQUAD | 5000 | 9.77E-01 | 2.30E-16 | 3.66E-11 | 2.30E-16 |
| SPARSQUR | 56 | 3.24E-09 | 1.69E-14 | ‡ | ‡ |
| LUKVLE12 | 2502 | 9.55E-01 | 6.33E-16 | 3.22E-09 | 1.63E-15 |
| MSQRTA | 64 | 3.69E-12 | 4.66E-15 | ‡ | ‡ |
| YATP1SQ | 352 | 2.10 | 9.44E-16 | 2.41E-09 | 9.44E-16 |

$nnz(row)$ denotes max number of entries in a row of $H$

‡ indicates error exceeds $10^{15}$

Conclude:

- Algorithm 1 has problems if rows are "dense".
- Algorithm 2 is unstable.

## How to overcome problems of simple approach?

Suppose $B^{(k)}$ has been permuted to form

$$\begin{pmatrix} B_{11}^{(k)} & B_{12}^{(k)} \\ B_{21}^{(k)} & B_{22}^{(k)} \end{pmatrix}, \quad \text{with} \quad B_{21}^{(k)} = (B_{12}^{(k)})^T.$$

- $B_{11}^{(k)}$ and $B_{22}^{(k)}$ are square symmetric matrices
- the $n_1 < n$ rows of $(B_{11}^{(k)} \; B_{12}^{(k)})$ are sparse
- remaining $n_2 = n - n_1 \ll n_1$ rows of $(B_{21}^{(k)} \; B_{22}^{(k)})$ considered "dense"

## Algorithm 3: Sparse-dense block parallel approach

- Use Algorithm 1 to compute sparse rows $(B_{11}^{(k)} \; B_{12}^{(k)})$ (parallel)
- Set $B_{21}^{(k)}$ by symmetry
- Use Algorithm 1 for small $n_2 \times n_2$ block $B_{22}^{(k)}$ (parallel)

## Example 3

$$B^{(k)} = \left(\begin{array}{cc|cc} b_{11}^{(k)} & 0 & b_{13}^{(k)} & b_{14}^{(k)} \\ 0 & 0 & b_{23}^{(k)} & b_{24}^{(k)} \\ \hline b_{31}^{(k)} & b_{32}^{(k)} & b_{33}^{(k)} & b_{34}^{(k)} \\ b_{41}^{(k)} & b_{42}^{(k)} & b_{43}^{(k)} & b_{44}^{(k)} \end{array}\right)$$

Rows 1 and 2 are sparse and rows 3 and 4 dense.
Linear system for dense row 3 is

$$\begin{pmatrix} s_3^{(k)} & s_4^{(k)} \\ s_3^{(k-1)} & s_4^{(k-1)} \end{pmatrix} \begin{pmatrix} b_{33}^{(k)} \\ b_{34}^{(k)} \end{pmatrix} = \begin{pmatrix} y_3^{(k)} \\ y_3^{(k-1)} \end{pmatrix} - \begin{pmatrix} s_1^{(k)} & s_2^{(k)} \\ s_1^{(k-1)} & s_2^{(k-1)} \end{pmatrix} \begin{pmatrix} b_{13}^{(k)} \\ b_{23}^{(k)} \end{pmatrix}$$

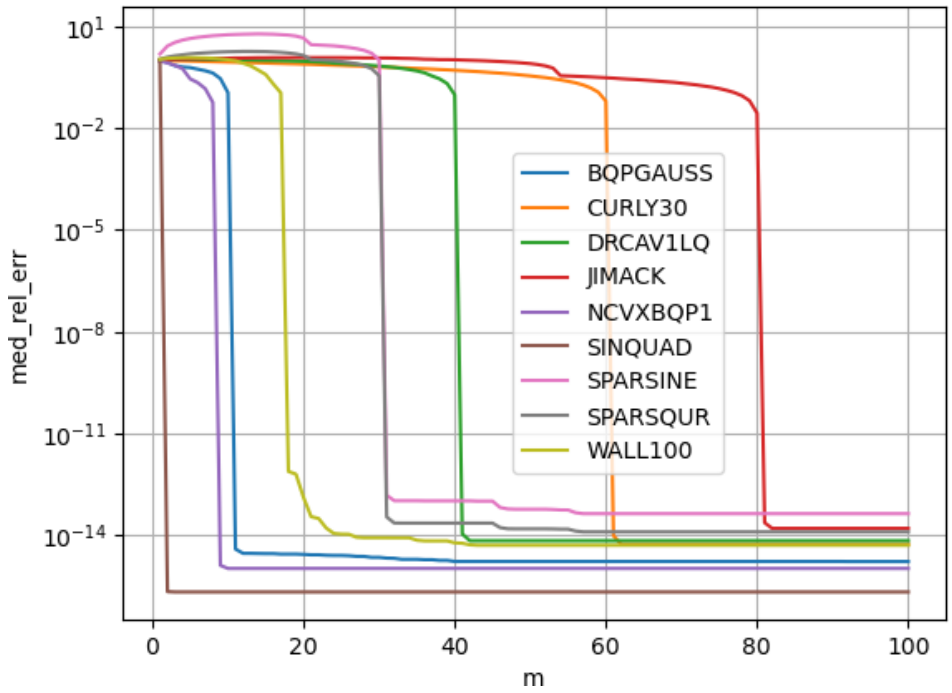
Similar expression for dense row 4.
Final value of entries (3,4) and (4,3) of $B^{(k)}$ is $(b_{34}^{(k)} + b_{43}^{(k)})/2$ (symmetrisation).

## Does this work? ($m = 100$)

| identifier | *max_rel_err* | *med_rel_err* | *serial* | *parallel* | *speedup* |
|---|---|---|---|---|---|
| CURLY30 | 5.41E-11 | 5.56E-15 | 4.55 | 0.34 | 13.23 |
| NCVXBQP1 | 3.15E-11 | 1.07E-15 | 0.53 | 0.05 | 10.06 |
| SINQUAD | 1.99E-11 | 2.17E-16 | 0.01 | 0.01 | 2.17 |
| SPARSQUR | 7.63E-10 | 1.28E-14 | 1.40 | 0.12 | 11.38 |
| LUKVLE12 | 4.48E-13 | 6.66E-16 | 0.04 | 0.01 | 4.78 |
| MSQRTA | 9.47E-13 | 2.66E-15 | 0.51 | 0.06 | 8.10 |
| YATP1SQ | 1.36E-11 | 9.17E-16 | 0.49 | 0.07 | 7.17 |

Times in seconds are on single core and 28 cores of a dedicated machine with 32 AMD Epyc 7502 CPUs clocked at 2.5GHz with 256 GB of RAM

Now vary number $m$ of past iterates. Plot is median relative error.




- If insufficient past data ($m$ too small) do not get accurate $H$
- Once $m$ sufficiently large, sharp transition to accurate $H$
- e.g, CURLY30 has a banded Hessian (bandwidth 61)

Can we do better?

## Algorithm 4: Recursive block parallel approach

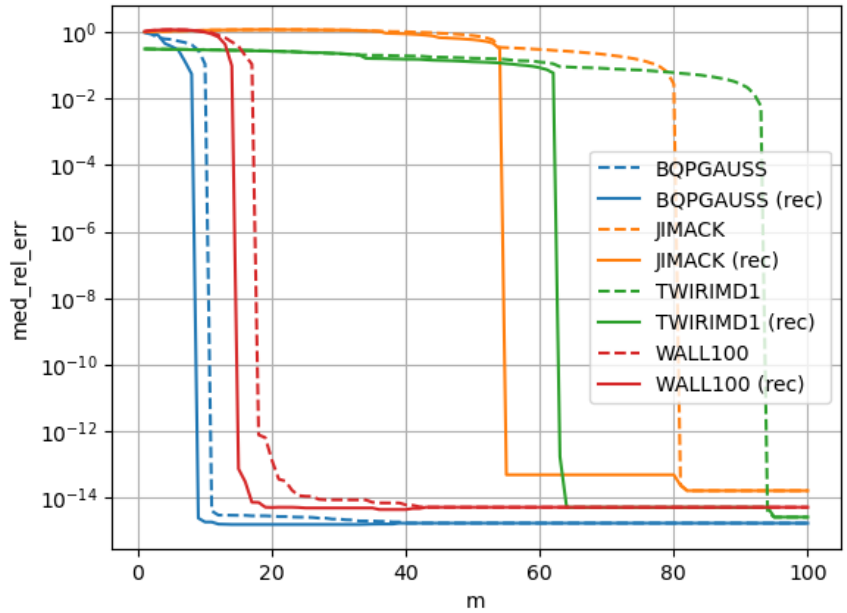
Apply block approach recursively.

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{pmatrix} \implies B_{22} = \begin{pmatrix} B_{11}^{(1)} & B_{12}^{(1)} \\ (B_{12}^{(1)})^T & B_{22}^{(1)} \end{pmatrix} \implies B_{22}^{(1)} = \begin{pmatrix} B_{11}^{(2)} & B_{12}^{(2)} \\ (B_{12}^{(2)})^T & B_{22}^{(2)} \end{pmatrix}$$

Rows in $(B_{11}\ B_{12})$ are sparse. Compute with Algorithm 1.

Then set $B_{12}^T$. Repeat process on $B_{22}$ (split it into sparse and dense rows).

Problems for which recursion is beneficial.




- Can get good improvment with smaller $m$
- e.g, JIMACK ($nnz(row) = 81$)

## Concluding remarks

- We have addressed the problem of approximating large sparse Hessian matrices.
- We have proposed new algorithms and derived one that is efficient in parallel and stable.
- We obtain accurate approximate Hessians provided there is sufficient past data.
- The methods are available in the Fortran module `sha`, with a C interface, as part of the open-source GALAHAD library.

# Thank you for listening.

Funding from EPSRC grant number EP/X032485/1.

*Approximating large-scale Hessian matrices using secant equations.*
J.M. Fowkes, N.I.M. Gould and J.A. Scott. Report STFC-P–2024-001.
To appear in ACM TOMS.