Numerical Optimisation: Large scale methods

Marta M. Betcke
m.betcke@ucl.ac.uk,
Kiko Rullan
f.rullan@cs.ucl.ac.uk

Department of Computer Science,
Centre for Medical Image Computing,
Centre for Inverse Problems
University College London

Lecture 9

Issues arising from large scale

- Hessian solve: Line search and trust region methods require factorisation of the Hessian. For large scale it is infeasible and has to be performed using large scale techniques such as sparse factorisations or iterative methods.

- Hessian computation and storage: Hessian approximations generated in quasi Newton methods are usually dense even if the true Hessian is sparse. Limited-memory variants have been developed, where the Hessian approximation can be stored using only few vectors (slow convergence). Approximated Hessians preserving sparsity.

- Special structure properties of the objective function like *partial separability* i.e. the function can be decomposed into a sum of simpler functions each depending only on a small subspace of $\mathbb{R}^n$.

Inexact Newton methods

Solve the Newton step system

$$\underbrace{\nabla^2 f_k}_{:=A} p_k^{\text{iN}} = \underbrace{-\nabla f_k}_{b}$$

using iterative method: CG or Lanczos with a modification to handle negative curvature.

Implementation can be done matrix free i.e. the Hessian does not need to be calculated or stored explicitly, we only require a routine which executes the Hessian matrix vector product.

Question: How does the inexact solve impact on the local convergence of the Newton methods?

Most of the termination rules for iterative methods are based on the residual

$$r_k = \nabla^2 f_k p_k^{\mathrm{iN}} + \nabla f_k,$$

where p_k^{iN} is the inexact Newton step.

Usually we terminate CG when

$$\|r_k\| \leq \eta_k \|\nabla f_k\|, \qquad \text{(iN-STOP)}$$

where $\{\eta_k\}$ is some sequence $0 < \eta_k < 1$.

For the moment we assume that step of length $\alpha_k = 1$ is taken i.e. globalisation strategies do not interfere with the inexact-Newton step.

Theorem: local convergence

Suppose $\nabla^2 f(x)$ exists and is continuous in the neighbourhood of a minimiser $x^\star$, with $\nabla^2 f(x^\star)$ positive definite.
Consider the inexact Newton iteration with step length $\alpha_k = 1$
$x_{k+1} = x_k + p_k$, with a starting point x_0 sufficiently close to $x^\star$, terminated with the stopping (iN-STOP) with $\eta_k \leq \eta$ for some constant $\eta \in [0, 1)$.

Then the sequence $\{x_k\}$ converges to $x^\star$ and satisfies

$$\|\nabla^2 f(x^\star)(x_{k+1} - x^\star)\| \leq \hat{\eta} \|\nabla^2 f(x^\star)(x_k - x^\star)\|,$$

for some constant $\hat{\eta} \, : \, \eta < \hat{\eta} < 1$.

Remark: This result provides convergence for $\{\eta_k\}$ bounded away from 1.

Proof idea convergence (superlinear):

Continuity of $\nabla^2 f(x)$ in a neighbourhood $\mathcal{N}(x^\star)$ of $x^\star$ implies

$$\nabla f(x_k) = \nabla^2 f(x^\star)(x_k - x^\star) + o(\|x_k - x^\star\|),$$

thus show instead $\|\nabla f(x_{k+1})\| \le \hat{\eta}\|\nabla f(x_k)\|$.

Continuity and positive definiteness of $\nabla^2 f(x)$ in $\mathcal{N}(x^\star)$ implies $\exists L \in \mathbb{R} > 0: \ \|\nabla^2 f(x_k)^{-1}\| \le L, \forall x_k \in \mathcal{N}(x^\star)$ and hence

$$\|p_k\| \le L(\|\nabla f(x_k)\| + \|r_k\|) \le 2L\|\nabla f(x_k)\|.$$

From Taylor theorem and continuity of $\nabla^2 f(x)$ in $\mathcal{N}(x^\star)$ we have

$$\begin{aligned}\nabla f(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k) p_k + \int_0^1 [\nabla^2 f(x_k + tp_k) - \nabla^2 f(x_k)] p_k \, dt \\ &= \nabla f(x_k) + \nabla^2 f(x_k) p_k + o(\|p_k\|) \\ &= \nabla f(x_k) - (\nabla f(x_k) - r_k) + o(\|\nabla f(x_k)\|) = r_k + o(\|\nabla f(x_k)\|)\end{aligned}$$

$$\|\nabla f(x_{k+1})\| \le \eta_k \|\nabla f(x_k)\| + o(\|\nabla f(x_k)\|) \le (\eta_k + o(1))\|\nabla f(x_k)\|$$

with $\eta_k = o(1)$, $\quad \le o(\|\nabla f(x_k)\|)$.

Proof idea convergence (quadratic):

Continuity of $\nabla^2 f(x)$ in a neighbourhood $\mathcal{N}(x^\star)$ of $x^\star$ implies

$$\nabla f(x_k) = \nabla^2 f(x^\star)(x_k - x^\star) + o(\|x_k - x^\star\|),$$

thus show instead $\|\nabla f(x_{k+1})\| \le \hat{\eta}\|\nabla f(x_k)\|$.

Continuity and positive definiteness of $\nabla^2 f(x)$ in $\mathcal{N}(x^\star)$ implies $\exists L \in \mathbb{R} > 0 : \ \|\nabla^2 f(x_k)^{-1}\| \le L, \forall x_k \in \mathcal{N}(x^\star)$ and hence

$$\|p_k\| \le L(\|\nabla f(x_k)\| + \|r_k\|) \le 2L\|\nabla f(x_k)\|.$$

From Taylor theorem and Lipschitz continuity of $\nabla^2 f(x)$ in $\mathcal{N}(x^\star)$

$$\begin{aligned}
\nabla f(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k) p_k + \int_0^1 [\nabla^2 f(x_k + tp_k) - \nabla^2 f(x_k)] p_k \, dt \\
&= \nabla f(x_k) + \nabla^2 f(x_k) p_k + \mathcal{O}(\|p_k\|^2) \\
&= \nabla f(x_k) - (\nabla f(x_k) - r_k) + \mathcal{O}(\|\nabla f(x_k)\|^2) = r_k + \mathcal{O}(\|\nabla f(x_k)\|^2)
\end{aligned}$$

with $\eta_k = \mathcal{O}(\|\nabla f(x_k)\|)$

$$\|\nabla f(x_{k+1})\| \le \eta_k \|\nabla f(x_k)\| + \mathcal{O}(\|\nabla f(x_k)\|^2) \le \mathcal{O}(\|\nabla f(x_k)\|^2).$$

Theorem: superlinear (quadratic) convergence

Suppose $\nabla^2 f(x)$ exists and is continuous in the neighbourhood of a minimiser $x^\star$, with $\nabla^2 f(x^\star)$ positive definite.
Let the sequence $\{x_k\}$ generated by the inexact Newton iteration with step length $\alpha_k = 1$, $x_{k+1} = x_k + p_k$ with stopping (iN-STOP) and $\eta_k \leq \eta$ for some constant $\eta \in [0, 1)$ and a starting point x_0 sufficiently close to $x^\star$, converge to $x^\star$.

Then the rate of convergence is superlinear if $\eta_k \to 0$.

If in addition $\nabla^2 f(x)$ is Lipschitz continuous for $x \in \mathcal{N}(x^\star)$ and $\eta_k = \mathcal{O}(\|\nabla f_k\|)$, then the convergence is quadratic.

Remark: To obtain superlinear convergence we can set e.g. $\eta_k = \min(0.5, \sqrt{\|\nabla f_k\|})$. The choice $\eta_k = \min(0.5, \|\nabla f_k\|)$ would yield quadratic convergence.

Line search Newton CG

Also called *truncated Newton method*. The key differences to standard Newton line search method:

- Solve the Newton step with CG with initial guess 0 and the termination criterium (iN-STOP) with the suitable choice of η_k, e.g. $\eta_k = \min(0.5, \sqrt{\|\nabla f_k\|})$ for superlinear convergence. Note, that if we are close enough to the solution the stopping tolerance decreases in each outer (line search) iteration.
- The inner CG iteration can be preconditioned.
- Away from the solution $x^\star$ the Hessian may not be positive definite. Therefore, we terminate CG whenever a direction of of non-positive curvature is generated $d_j^T \nabla^2 f_k d_j \leq 0$. This guarantees that the produced search direction is a descent direction and preserves the fast pure Newton convergence rate provided $\alpha_k = 1$ is used whenever it satisfies the acceptance criteria.

Weakness: Performance when Hessian is nearly singular.

Algorithm 7.1 (Line Search Newton–CG).

```
Given initial point $x_0$;
for $k = 0, 1, 2, \ldots$
        Define tolerance $\epsilon_k = \min(0.5, \sqrt{\|\nabla f_k\|})\|\nabla f_k\|$;
        Set $z_0 = 0, r_0 = \nabla f_k, d_0 = -r_0 = -\nabla f_k$;
        for $j = 0, 1, 2, \ldots$
                if $d_j^T B_k d_j \le 0$
                        if $j = 0$
                                return $p_k = -\nabla f_k$;
                        else
                                return $p_k = z_j$;
                Set $\alpha_j = r_j^T r_j / d_j^T B_k d_j$;
                Set $z_{j+1} = z_j + \alpha_j d_j$;
                Set $r_{j+1} = r_j + \alpha_j B_k d_j$;
                if $\|r_{j+1}\| < \epsilon_k$
                        return $p_k = z_{j+1}$;
                Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$;
                Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$;
        end (for)
        Set $x_{k+1} = x_k + \alpha_k p_k$, where $\alpha_k$ satisfies the Wolfe, Goldstein, or
                Armijo backtracking conditions (using $\alpha_k = 1$ if possible);
end
```

Trust region Newton CG

Use a special CG variant to solve the quadratic trust region model problem

$$\min_{p\in\mathbb{R}^n} m_k(p) = f_k + p^{\mathrm{T}}\nabla f_k + \frac{1}{2}p^{\mathrm{T}}\nabla^2 f_k p \quad \text{subject to } \|p\| \le \Delta_k.$$

Modifications:

- Use the termination criterium (IN-STOP) as in line search variant with a suitable choice of η_k,
  e.g. $\eta_k = \min(0.5, \sqrt{\|\nabla f_k\|})$ for superlinear convergence.
- If CG generates direction of non-positive curvature
  i.e. $d_j^{\mathrm{T}}\nabla^2 f_k d_j \le 0$, stop and return $p_k = z_j + \tau d_j$ which minimises $m_k(p_k)$ along d_j and satisfies $\|p_k\| = \Delta_k$.
- If the current iterate violates the trust region constraint
  i.e. $\|z_{j+1}\| \ge \Delta_k$, stop and return $p_k = z_j + \tau d_j$, $\tau \ge 0$ which satisfies $\|p_k\| = \Delta_k$.

Algorithm 7.2 (CG–Steihaug).

```
Given tolerance $\epsilon_k > 0$;
Set $z_0 = 0, r_0 = \nabla f_k, d_0 = -r_0 = -\nabla f_k$;
if $\|r_0\| < \epsilon_k$
        return $p_k = z_0 = 0$;
for $j = 0, 1, 2, \ldots$
        if $d_j^T B_k d_j \le 0$
                Find $\tau$ such that $p_k = z_j + \tau d_j$ minimizes $m_k(p_k)$ in (4.5)
                        and satisfies $\|p_k\| = \Delta_k$;
                return $p_k$;
        Set $\alpha_j = r_j^T r_j / d_j^T B_k d_j$;
        Set $z_{j+1} = z_j + \alpha_j d_j$;
        if $\|z_{j+1}\| \ge \Delta_k$
                Find $\tau \ge 0$ such that $p_k = z_j + \tau d_j$ satisfies $\|p_k\| = \Delta_k$;
                return $p_k$;
        Set $r_{j+1} = r_j + \alpha_j B_k d_j$;
        if $\|r_{j+1}\| < \epsilon_k$
                return $p_k = z_{j+1}$;
        Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$;
        Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$;
end (for).
```

The initialisation $z_0 = 0$ is crucial:

- Whenever $\|r_k\| \geq \varepsilon_k$, the algorithm terminates at a point p_k for which $m_k(p_k) \leq m_k(p_k^C)$ that is when the reduction in the model is at least that of the Cauchy point.
    - If $d_0^{\mathrm{T}} B_k d_0 = \nabla f_0^{\mathrm{T}} B_0 \nabla f_0 \leq 0$, the first **if** is activated and the algorithm returns the Cauchy point $p = -(\Delta_0/\|\nabla f_0\|)\nabla f_0$.
    - Otherwise, the algorithm defines

$$z_1 = \alpha_0 d_0 = \frac{r_0^{\mathrm{T}} r_0}{d_0^{\mathrm{T}} B_k d_0} d_0 = -\frac{\nabla f_0^{\mathrm{T}} \nabla f_0}{\nabla f_0^{\mathrm{T}} B_0 \nabla f_0} \nabla f_0.$$

        - If $\|z_1\| < \Delta_0$ then z_1 is exactly the Cauchy point. Subsequent steps ensure that the final p_k satisfies $m_k(p_k) \leq m_k(z_1)$.
        - When $\|z_1\| \geq \Delta_0$, the second **if** is activated and the algorithm terminates at the Cauchy point.

  Therefore, it is globally convergent.

- $\|z_{k+1}\| > \|z_k\| > \cdots > \|z_1\|$ as a consequence of the initialisation $z_0 = 0$. Thus we can stop as soon as the boundary of trust region has been reached, because no further iterates giving a lower value of m_k will lie inside the trust region.

- Preconditioning can be used, but requires change of trust region definition, which can be reformulated in the standard form in terms of a variable $\hat{p} = Dp$ and modified $\hat{g}_k = D^{-\mathrm{T}} \nabla f_k$ and $\hat{B}_k = D^{-\mathrm{T}} (\nabla^2 f_k) D^{-1}$. Of particular interest is incomplete Cholesky factorisation (Algorithm 7.3 in Nocedal and Wright).
- The limitation of the algorithm is that it accepts any direction of negative curvature, even if this direction gives insignificant reduction in the model. To improve performance, CG can be replaced by Lanczos method (which can be seen as generalisation of CG which works for indefinite system albeit is more computationally expensive) for which techniques from exact trust region can be applied to compute a direction to quickly move away from stationary points which are not minimisers.

Limited memory quasi-Newton methods

Recall the BFGS formula

$$H_{k+1} = (I - \frac{s_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k}) H_k (I - \frac{y_k s_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k}) + \frac{s_k s_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k} \qquad \text{(BFGS)}$$

with $s_k = x_{k+1} - x_k = \alpha_k p_k$, $\quad y_k = \nabla f_{k+1} - \nabla f_k$. Application of BFGS Hessian approximation can be efficiently implemented storing the list of vector pairs (s_k, y_k).

The limited memory version can be obtained by restricting the total number of vectors used to construct the Hessian approximation to the last $m \ll n$. After the mth update, the oldest pair in the list makes space for the new pair.

Same strategy can be applied to the other quasi-Newton schemes (including updating B_k for use with e.g. trust region methods rather than line search methods which require H_k).

Application: large, non-sparse Hessians.
Convergence: often linear convergence rate.

Theoretical connection to CG methods

Consider the *memoryless* BFGS

$$H_{k+1} = (I - \frac{s_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k})(I - \frac{y_k s_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k}) + \frac{s_k s_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k}$$

i.e. the previous Hessian is reset to identity, $H_k = I$.
If the memory less BFGS is applied in conjunction with an exact line search i.e. $p_k^{\mathrm{T}} \nabla f_{k+1} = 0$ for all k. We then obtain

$$p_{k+1} = -H_{k+1}\nabla f_{k+1} = -\nabla f_{k+1} + \frac{y_k^{\mathrm{T}} \nabla f_{k+1}}{y_k^{\mathrm{T}} p_k} p_k,$$

which is exactly the Hestens-Stiefel formula, which reduces to Polak-Ribiere when $\nabla f_{k+1}^{\mathrm{T}} p_k = 0$

$$\beta_{k+1}^{HS} = \frac{\nabla f_{k+1}^{\mathrm{T}}(\nabla f_{k+1} - \nabla f_k)}{p_k^{\mathrm{T}}(\nabla f_{k+1} - \nabla f_k)}, \quad \beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^{\mathrm{T}}(\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^{\mathrm{T}} \nabla f_k}.$$

Compact representation of BFGS update

Let B_0 be symmetric positive definite and assume that the vector pairs $\{s_i, y_i\}_{i=0}^{k-1}$ satisfy $s_i^{\mathrm{T}} y_i > 0$. Applying k BFGS updates with these vector pairs to B_0 yields

$$B_k = B_0 - \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T B_0 \\ Y_k^T \end{bmatrix}$$

where S_k and Y_k are the $n \times k$ matrices defined by

$$S_k = [s_0, \ldots, s_{k-1}], \qquad Y_k = [y_0, \ldots, y_{k-1}],$$

while L_k and D_k are the $k \times k$ matrices

$$(L_k)_{i,j} = \begin{cases} s_{i-1}^T y_{j-1} & \text{if } i > j, \\ 0 & \text{otherwise,} \end{cases}$$

$$D_k = \operatorname{diag}\left[s_0^T y_0, \ldots, s_{k-1}^T y_{k-1}\right].$$

- In limited memory version we replace the columns or diagonal entries in the matrices cyclically (keeping m last columns).
- Since the dimension of the middle matrix is small, the factorisation cost is negligible.
- Cost of an update: $2mn + \mathcal{O}(m^3)$
- Cost of $B_k v$: $(4m+1)n + \mathcal{O}(m^3)$, (for $B_0 = \delta_k I$)
- This approximation can be used in trust region methods for unconstrained problems, but also in methods for constrained optimisation.
- Similar compact representation can be derived for H_k
- Compact representation can also be derived for SR-1

$$B_k = B_0 + (Y_k - B_0 S_k)(D_k + L_k + L_k^{\mathrm{T}} - S_k^{\mathrm{T}} B_0 S_k)^{-1}(Y_k - B_0 S_k)^{\mathrm{T}}$$

with S_k, Y_k, D_k, L_k as before. The inverse formula for H_k can be obtained by swapping $B \leftrightarrow H$, $s \leftrightarrow y$, however limited memory SR-1 can be less effective than BFGS.

Sparse quasi-Newton updates

We require the quasi-Newton approximation to the Hessian B_k to has the same (or similar) sparsity pattern as the true Hessian. Suppose that we know which components of the Hessian are nonzero

$$\Omega = \left\{(i,j) : [\nabla^2 f(x)]_{ij} \neq 0 \text{ for some point } x \text{ in the domain of } f\right\},$$

and suppose that the current approximation B_k mirrors this sparsity structure. Such sparse update can be obtained as a solution of the following quadratic program

$$\begin{aligned} \min_B \quad & \|B - B_k\|_F^2 = \sum_{(i,j)\in\Omega} [B_{ij} - (B_k)_{ij}]^2, \\ \text{subject to} \quad & Bs_k = y_k,\ B = B^{\mathrm{T}},\ B_{ij} = 0\ \forall (i,j) \notin \Omega. \end{aligned}$$

It can be shown that the solution of this problem can be obtained solving an $n \times n$ linear system with sparsity pattern Ω. B_{k+1} is not guaranteed to be positive definite. The new B_{k+1} can be used within a trust region.

Unfortunately, this approach has several drawbacks, it is not scale invariant under linear transformations and the performance is disappointing. The fundamental weakness is that the closeness in Frobenius norm is an inadequate model and the produced approximations can be poor.

An alternative approach is to relax the secant equation making sure that it is approximately satisfied at the m last steps (as opposed to holding strictly in the last step) and solve

$$\begin{aligned} \min_B \quad & \|BS_k - Y_k\|_F^2, \\ \text{subject to} \quad & B = B^{\mathrm{T}},\ B_{ij} = 0\ \forall (i,j) \notin \Omega, \end{aligned}$$

with S_k, Y_k containing the last m of s_i, y_i, respectively.

This convex optimisation problem has a solution but it is not easy to compute. Furthermore, it can produce singular and poorly conditioned Hessian approximations. Even though it frequently outperforms the previous approach, its performance is still not impressive for large scale problems.

Partially separable functions

An unconstrained optimisation problem is **separable** if the objective function $f : \mathbb{R}^n \to \mathbb{R}$ can be decomposed in a sum of independent functions e.g.

$$f(x) = f_1(x_1, x_3) + f_2(x_2, x_4, x_6) + f_3(x_5).$$

The optimal value can be found optimising each function independently, which is in general much less expensive.

In many large scale problems the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is not separable but it still can be written as a sum of simpler component functions. Each such component has the property that it only changes in a small number of directions while for other directions is remains constant. We call such functions **partially separable**.

All functions which have a sparse Hessian are partially separable, but there are many partially separable functions with dense Hessians. Partial separability allows for economical representation and effective quasi-Newton updating.

Consider an objective function $f : \mathbb{R}^n \to \mathbb{R}$

$$f(x) = \sum_{i=1}^{\ell} f_i(x),$$

where each f_i depends only on a few components of x. For such f_i, its gradient and Hessian contain only few non-zeros.

For the function f we have by linearity of differentiation

$$\nabla f(x) = \sum_{i=1}^{\ell} \nabla f_i(x), \quad \nabla^2 f(x) = \sum_{i=1}^{\ell} \nabla^2 f_i(x)$$

thus we can maintain an quasi-Newton approximation to each individual component Hessian $\nabla^2 f_i(x)$ instead of approximating the entire Hessian $\nabla^2 f(x)$.

Example: partially separable approximation

Consider a partially separable objective function

$$f(x) = \underbrace{(x_1 - x_3^2)^2}_{f_1(x)} + \underbrace{(x_2 - x_4^2)^2}_{f_2(x)} + \underbrace{(x_3 - x_2^2)^2}_{f_3(x)} + \underbrace{(x_4 - x_1^2)^2}_{f_4(x)}.$$

Each f_i depends on two components only, all have the same form.

Denote

$$x^{[1]} = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix}, \ U_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \ x^{[1]} = U_1 x, \ \phi_1(z_1, z_2) = (z_1 - z_2^2)^2.$$

Then $f_1(x) = \phi(U_1 x)$ and using chain rule we obtain

$$\nabla f_1(x) = U_1^{\mathrm{T}} \nabla \phi_1(U_1 x), \quad \nabla^2 f_1(x) = U_1^{\mathrm{T}} \nabla^2 \phi_1(U_1 x) U_1.$$

For the Hessians $\nabla^2\phi_1$ and $\nabla^2 f_1$ we have

$$\nabla^2\phi_1(U_1x) = \begin{bmatrix} 2 & -4x_3 \\ -4x_3 & 12x_3^2 - 4x_1 \end{bmatrix}, \quad \nabla^2 f_1(x) = \begin{bmatrix} 2 & 0 & -4x_3 & 0 \\ 0 & 0 & 0 & 0 \\ -4x_3 & 0 & 12x_3^2 - 4x_1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Idea: maintain quasi-Newton approximation $B^{[1]}$ to 2×2 Hessian $\nabla^2\phi_1$ and lift it up to $\nabla^2 f_1$.

After a step from x_k to x_{k+1}

$$s_k^{[1]} = x_{k+1}^{[1]} - x_k^{[1]}, \quad y_k^{[1]} = \nabla\phi_1(x_{k+1}^{[1]}) - \nabla\phi_1(x_k^{[1]}),$$

and we use BFGS or SR-1 updating to obtain the new approximation $B_{k+1}^{[1]}$ of the small, dense Hessian $\nabla^2\phi_1$ and we lift it back using

$$\nabla^2 f_1(x) \approx U_1^{\mathrm{T}} B_{k+1}^{[1]} U_1.$$

We do the same for all component functions and we obtain

$$\nabla^2 f \approx B = \sum_{i=1}^{\ell} U_i^{\mathrm{T}} B^{[1]} U_i.$$

- The approximated Hessian may be used in trust region algorithm, obtaining an approximate solution to

$$B_k p_p = -\nabla f_k.$$

  B_k does not need to be assembled explicitly but conjugate gradient method can be used and the products $B_k v$ can be performed directly using matrices U_i and $B^{[i]}$.

- This approach is particularly useful for large number of variables with very small dependence of component functions. Then each respective component Hessian can be much faster approximated by the iterative method (small problem requires few directions) and the so obtained full Hessian approximation is usually much better than one obtained by a quasi-Newton method applied to the problem ignoring the partially separable structure (large Hessian requires a lot of directions to approximate the curvature).

- It is not always possible for BFGS to update the partial Hessian $B^{[1]}$, as the curvature condition $(s^{[1]})^{\mathrm{T}} y^{[1]} > 0$ may not be satisfied even if the full Hessian is at least positive semidefinite. This can be overcome applying SR-1 update to the component Hessians, which proved effective in practice.
- The limitation of this quasi-Newton approach is the cost of computing the step, which is comparable to the cost of Newton step, thus it may be beneficial to actually take the Newton step.
- Another problem is the difficulty of identifying the partially separable structure of a function. The performance of quasi-Newton methods is satisfactory provided that we find the *finest* partially separable decomposition.