Partially separable structure (PSS) May 20, 2021

Paul Raynaud

May 20, 2021

Polytechnique Montréal, Grenoble INP

Problem statement

Problem of interest:

$$\min_{x \in \mathbb{R}^n} f(x) \qquad f(x) := \sum_{i=1}^{N} f_i(x)$$

- large problems $n > 10^3$
- $f_i : \mathbb{R}^n \to \mathbb{R}$ does not depend on all of x
- $f_i \in \mathscr{C}^2$, $i = 1, .., N$

Example:

$$\min_{x \in \mathbb{R}^n} f_1(x_1, x_2) + f_n(x_{n-1}, x_n) + \sum_{i=2}^{n-1} f_i(x_{i-1}, x_i, x_{i+1})$$

Definition

The linear operator U_i gives the (linear combination of) variables used by f_i:

$$\begin{aligned}
f(x) &= \textstyle\sum_{i=1}^N \widehat{f}_i(U_i x) \\
\nabla f(x) &= \textstyle\sum_{i=1}^N U_i^\top \nabla \widehat{f}_i(U_i x) \\
\nabla^2 f(x) &= \textstyle\sum_{i=1}^N U_i^\top \nabla^2 \widehat{f}_i(U_i x) U_i \\
\nabla^2 f(x) \approx B &= \textstyle\sum_{i=1}^N U_i^\top \widehat{B}_i U_i
\end{aligned}$$

- $\widehat{f}_i : \mathbb{R}^{n_i} \to \mathbb{R}$ an element function
- $U_i \in \mathbb{R}^{n_i \times n}$ usually a linear operator far more efficient than a matrix
- $\widehat{B}_i \in \mathbb{R}^{n_i \times n_i}, i = 1, \ldots, N$
- If $\max\limits_{i=\{1,\ldots,N\}} n_i \ll n$, store $\{\widehat{B}_i\}_{i=1}^N$ requires (much) less memory than B

Theorem (Griewank and Toint [1982a])
Every problem having a sparse hessian is partially separable.

Motivation for studying the PSS

The PSS allows partitioned QN updates (PQN) (Griewank and Toint [1982b])

$$B = \sum_{i=1}^{N} B_i = \sum_{i=1}^{N} U_i^\top \widehat{B}_i U_i$$

- Apply QN update to each $\widehat{B}_i$ using $U_i s$ and $\nabla \widehat{f}_i(U_i(x+s)) - \nabla \widehat{f}_i(U_i x)$
- $\sum_{i=1}^{N} B_i$ still satisfies secant equation
- Advantages:
  - does not increase memory requirements $\{\widehat{B_i}\}_{i=1}^{N}$ ($\neq$ standard QN)
  - keep the sparsity of B ($\neq$ L-BFGS)
  - fully parallelizable: each $\widehat{B}_i$ update is independent: $(U_i s, \widehat{y}_i)$ such that $\widehat{y}_i = \nabla \widehat{f}_i(U_i(x+s)) - \nabla \widehat{f}_i(U_i x)$
  - rank update $\gg$ 1 or 2

Towards a complete optimization method

A trust-region method or a linesearch framework around the PQN update leads us to solve a partitioned linear system at every iteration:

- Conjugate gradient (CG)
  - require matrix-vector products: $Bv = \left(\sum U_i^\top \widehat{B}_i U_i\right) v$
  - can compute $\widehat{B}_i U_i v$ in parallel and assemble with $U_i^\top$
- (multi-)frontal factorization (Conn et al. [1994])
  - Cholesky factorization dedicated to partitioned matrix
  - $\{U_i\}_{i=1}^N$ provide the sparsity of B
  - the permutation applied to the matrix is **critical**: front size, filling, parallelizable blocs
- partitioned trust-region method (Conn et al. [1996])

Efficient derivatives computation

Reduce $f(x) = \sum_{i=1}^{N} \widehat{f}_i(U_i x)$ evaluation required to compute ∇f from $\{\nabla \widehat{f}_i\}_{i=1}^{N}$ in case every $\widehat{f}_i(x)$ are evaluated at once and by using the structure $\{U_i\}_{i=1}^{N}$.

$$f(x) = \sum_{i=1}^{5} f_i(x) = 1^\top F(x) = \widehat{f}_1(x_1, x_3) + \widehat{f}_2(x_1, x_4) + \widehat{f}_3(x_2, x_3) + \widehat{f}_4(x_2, x_4) + \widehat{f}_5(x_3)$$

$$F(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \\ f_4(x) \\ f_5(x) \end{pmatrix}, \quad \nabla F = \begin{pmatrix} \square & & \triangle & \\ \square & & & \diamond \\ & \Diamond & \triangle & \\ & \Diamond & & \diamond \\ & & \triangle & \end{pmatrix}, \quad S_c = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

If ∇F is dense the seed S to compute ∇F is I_4 implying 4 f evaluations. The PSS $\{U_i\}_{i=1}^{N}$ induce graph structure whose a proper coloring define the compressed seed S_c implying 2 f evalutions.

Automatic differentiation (AD)

Compute the derivative of a numerical procedure $f : \mathbb{R}^n \to \mathbb{R}^m$

- Forward mode
  - more efficient than reverse if $m > n$
  - memoryless method
- Reverse mode
  - more efficient than forward if $m < n$
  - must build a tape of the numerical procedure.

AD and PSS

- If every $\widehat{f}_i$ is available and evaluate at once a similar procedure to compressed seed may be used (Bischof et al. [1997])
- If each $\widehat{f}_i$ is available individually:
    - forward mode is more efficient since $n_i \ll n$
    - the tape of each $\widehat{f}_i$ is much smaller than f (smaller expression tree)
    - in practice, $\widehat{f}_i = \widehat{f}_j$ allowing to reduce the number of tapes needed
- Hessian-vector products $\nabla^2 \widehat{f}_i(U_i x) U_i v$ combine both approaches and their properties in PSS. It allows a complete parallel procedure to compute $\nabla^2 f(x) v$ from $\sum_{i=1}^{N} U_i^\top \nabla^2 \widehat{f}_i(U_i x) U_i v$

Other uses of PSS in the literature

- Dedicated crossover operator, the key of genetic algorithms (Durand and Alliot [1998])
- Specific to DFO:
    - Interpolations based on the knowledge of $\{\widehat{f}_i\}_{i=1}^N$
        - By interpolating each $\widehat{f}_i$, $\approx n_i^2$ points instead of $\approx n^2$
        - Reducing the $\{\widehat{f}_i\}_{i=1}^N$ evaluations depending the structure to obtain those n_i^2 points
    - Dedicated efficient procedure to recompute $f, \nabla f$ if $x_{k+1} - x_k$ is sparse, only the $\widehat{f}_i, \nabla \widehat{f}_i$ impact must be recompute
- Brute Force Optimizer (BFO): (Porcelli and Toint [2021])

Limitations

- Problem structure must be explicited by the modeler
- ⚠if $\sum n_i^2 \geq n^2$: not applicable in large scale, require more space and computation than BFGS, ex: $f(x) = \sum_{i=1}^{n} \widehat{f_i}(x_1, x_2, ..., x_i)$
  - Method to find a new basis to increase the sparsity of the problem Kim et al. [2009]

Historically

- Study of PSS is about 40 years old Griewank and Toint [1982a]
- During the last 40 years, work mainly done by Conn, Gould and Toint
- Resulting LANCELOT a Fortran software using the SIF format
- AMPL (commercial software) also uses the PSS and detects it automatically.

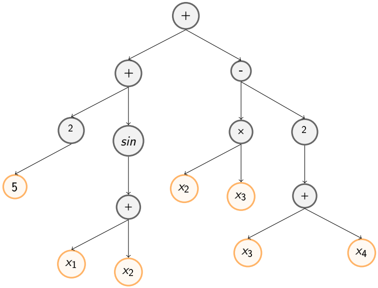
My work

- Provide modern software to detect PSS automatically:
  - Assess convexity of the f_i automatically
  - Construct new optimization methods that exploit PSS
  - 4 julia modules
  - Make it easily usable ($\neq$ LANCELOT)
- Survey on PSS

Module CalculusTreeTools.jl

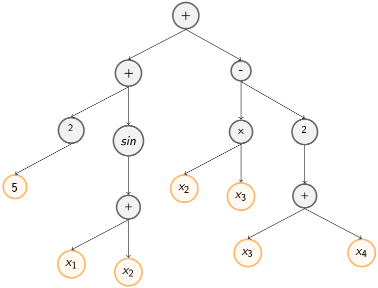
- Detect PSS $(\{\widehat{f}_i\}_{i=1}^N, \{U_i\}_{i=1}^N)$ automatically from f
- Automatic strict convexity detection and bounds propagation
- Interfaced to JuMP, NLPModelJuMP, ADNLPModel

Example



$f_1(x) = 5^2$	$[5]$	*constant*	*non strictly convex*
$f_2(x) = sin(x_1 + x_2)$	$[-1,1]$	*nonlinear*	*non strictly convex*
$f_3(x) = x_2 \times x_3$	$[-\infty,\infty]$	*quadratic*	*non strictly convex*
$f_4(x) = -(x_3 + x_4)^2$	$[0,\infty]$	*quadratic*	*non strictly convex*

Example



$$U_1 = 0 \quad U_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \; or \; U_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix} \quad U_3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$U_4 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \; or \; U_4 = \begin{pmatrix} 0 & 0 & 1 & 1 \end{pmatrix}$$

Module PartiallySeparableNLPModel.jl

- define the algorithm structures around PSS
- Test problem Rosenbrock function ($\mathbb{R}^n \rightarrow \mathbb{R}$)

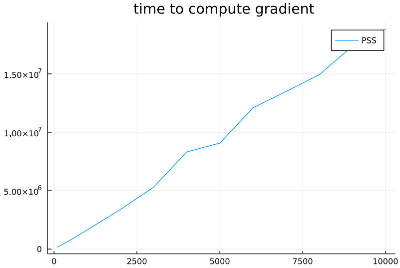


Figure 1: PSF gradient t/n

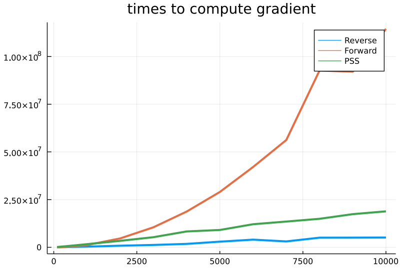


Figure 2: comparison ton AD t/n

- Trust-region methods using partitioned-QN solved by CG
- 40 PS problems of size $n = 1000$

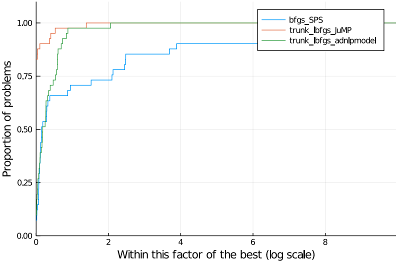


Figure 3: time

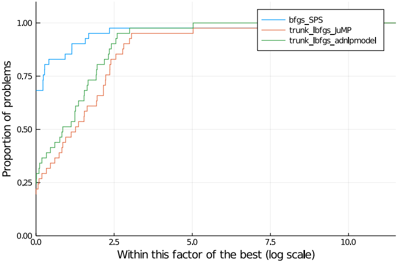


Figure 4: obj+5grad

Module PartitionedStructure.jl

- Dedicated to partitioned structured: vectors/matrices. Also somes specificity about PSS.
- Multi-frontal factorization implementation

Improving axes

Currently a trust region using a P-BFGS update must solve at each iterate the partitioned linear system:

$$\begin{gathered} Ax = b \\ \sum_i U_i^\top \widehat{A}_i U_i x = \sum_i U_i^\top \widehat{b}_i \\ \widehat{A}_i = \widehat{B}_i, \widehat{b}_i = -\nabla \widehat{f}_i, x = s \end{gathered}$$

The complexity of the whole method:

- PQN: update $\{\widehat{B}_i\}_{i=1}^N$ (fully parallelizable, depend of n_i)
- TR management is constant
- Solving the partitioned linear system: CG is state of the art

Method to solve a partitonned linear system

The following properties must hold:

- Do not form B
- Be parallel
- To use it with a trust region an approximate solution is enough.
- The solution must be a descent direction

The ideal would be an iterative method that iteratively check TR constraint (similar to CG).

An other way to parallelize

In **completely** separable case solving each $\widehat{A_i}\widehat{x}^i = \widehat{b}_i$, $x^i \in \mathbb{R}^{n_i}$ solve $Ax = b$.

$$\begin{pmatrix} \widehat{A}_1 & 0 \\ 0 & \widehat{A}_2 \end{pmatrix} \begin{pmatrix} x \end{pmatrix} = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix}$$

- Plan to form a solution x from $\{\widehat{x}^i\}_{i=1}^2$ such that $\widehat{A}_i\widehat{x}^i = \widehat{b}_i$
- Consequently each $\widehat{A}_i x^i = \widehat{b}_i$ may be solve in parallel.

$$x = \begin{pmatrix} \widehat{x}^1 \\ \widehat{x}^2 \end{pmatrix} \tag{1}$$

The cas of two overlapping blocs

Suppose $\widehat{A}_1, \widehat{A}_2 \in \mathbb{R}^{n_1 \times n_1}, \mathbb{R}^{n_2 \times n_2}$ such that $A \in \mathbb{R}^{n \times n}$.

$$Ax = \begin{pmatrix} \widehat{A}_{1_{1,1}} & \widehat{A}_{1_{1,2}} & 0 \\ \widehat{A}_{1_{2,1}} & \widehat{A}_{1_{2,2}} + \widehat{A}_{2_{1,1}} & \widehat{A}_{2_{1,2}} \\ 0 & \widehat{A}_{2_{2,1}} & \widehat{A}_{2_{2,2}} \end{pmatrix} x = \begin{pmatrix} \widehat{b}_{1_1} \\ \widehat{b}_{1_2} + \widehat{b}_{2_1} \\ \widehat{b}_{2_2} \end{pmatrix} = b$$

Suppose $\widehat{x}^1, \widehat{x}^2$ such that $\widehat{A}_p \widehat{x}^p = \widehat{b}_p$, $p = 1, 2$ and an approximation $x^?$ of x^* such that:

$$x^? = \begin{pmatrix} \widehat{x}_1^1 \\ \widehat{x}^? \\ \widehat{x}_2^2 \end{pmatrix} \quad x^1 = \begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix} \quad x^2 = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}$$

Consequently $Ax^? - b$

$$
\begin{aligned}
\widehat{A}_{p_{1,1}}\widehat{x}_1^p + \widehat{A}_{p_{1,2}}\widehat{x}_2^p &= \widehat{b}_{p_1} \\
\widehat{A}_{p_{2,1}}\widehat{x}_1^p + \widehat{A}_{p_{2,2}}\widehat{x}_2^p &= \widehat{b}_{p_2}
\end{aligned}
$$

Replace $\widehat{x}^p$ by $U_i x^?$:

$$
\begin{aligned}
\widehat{A}_{1_{1,1}}\widehat{x}_1^1 + \widehat{A}_{1_{1,2}}\widehat{x}_2^1 + \widehat{A}_{1_{1,2}}(\widehat{x}^? - \widehat{x}_2^1) &= \widehat{b}_{1_1} \\
\underbrace{\widehat{A}_{1_{1,1}}\widehat{x}_1^1 + \widehat{A}_{1_{1,2}}\widehat{x}_2^1 - \widehat{b}_{1_1}}_{=0} + \widehat{A}_{1_{1,2}}(\widehat{x}^? - \widehat{x}_2^1) &= 0 \\
\widehat{A}_{1_{2,1}}\widehat{x}_1^1 + \widehat{A}_{1_{2,2}}\widehat{x}_2^1 + \widehat{A}_{1_{2,2}}(\widehat{x}^? - \widehat{x}_2^1) &= \widehat{b}_{1_2} \\
\underbrace{\widehat{A}_{1_{2,1}}\widehat{x}_1^1 + \widehat{A}_{1_{2,2}}\widehat{x}_2^1 - \widehat{b}_{1_2}} + \widehat{A}_{1_{2,2}}(\widehat{x}^? - \widehat{x}_2^1) &= 0 \\
\widehat{A}_{2_{1,1}}\widehat{x}_1^2 + \widehat{A}_{2_{2,1}}\widehat{x}_2^2 + \widehat{A}_{2_{2,1}}(\widehat{x}^? - \widehat{x}_1^2) &= \widehat{b}_{2_1} \\
\underbrace{\widehat{A}_{2_{1,1}}\widehat{x}_1^2 + \widehat{A}_{2_{2,1}}\widehat{x}_2^2 - \widehat{b}_{2_1}} + \widehat{A}_{2_{2,1}}(\widehat{x}^? - \widehat{x}_1^2) &= 0 \\
\widehat{A}_{2_{2,1}}\widehat{x}_1^2 + \widehat{A}_{2_{2,2}}\widehat{x}_2^2 + \widehat{A}_{2_{1,1}}(\widehat{x}^? - \widehat{x}_1^2) &= \widehat{b}_{2_2} \\
\underbrace{\widehat{A}_{2_{2,1}}\widehat{x}_1^2 + \widehat{A}_{2_{2,2}}\widehat{x}_2^2 - \widehat{b}_{2_2}} + \widehat{A}_{2_{1,1}}(\widehat{x}^? - \widehat{x}_1^2) &= 0
\end{aligned}
$$

A new problem

The residual $Ax^? - b$ is the following:

$$Ax^? - b = -\left(\begin{array}{c} \widehat{A}_{1_{1,2}}(\widehat{x}^? - \widehat{x}_2^1) \\ \widehat{A}_{1_{2,2}}(\widehat{x}^? - \widehat{x}_2^1) + \widehat{A}_{2_{1,1}}(\widehat{x}^? - \widehat{x}_1^2) \\ \widehat{A}_{2_{2,1}}(\widehat{x}^? - \widehat{x}_1^2) \end{array} \right)$$

This equation link $x^?, \widehat{x}^?$ and a approximate solution $Ax = b$ (ie $Bs = -g$). We would like to minimize the residual $Ax^? - b$; depending only of $\widehat{x}^?$.

Remark: The optimum of this problem $Ax^? - b$ may be not null since the approximate $x^?$ is arbitrary.

Direct problem

$$\min_{\widehat{x}^? \in \mathbb{R}^{n_{inter}}} \| \underbrace{\begin{pmatrix} \widehat{A}_{1_{1,2}}(\widehat{x}^? - \widehat{x}_2^1) \\ \widehat{A}_{1_{2,2}}(\widehat{x}^? - \widehat{x}_2^1) + \widehat{A}_{2_{1,1}}(\widehat{x}^? - \widehat{x}_1^2) \\ \widehat{A}_{2_{2,1}}(\widehat{x}^? - \widehat{x}_1^2) \end{pmatrix}}_{\in \mathbb{R}^n} \|$$

- Problem dimension: $n = n_1 + n_2 - n_{inter}$
- Variable dimension: n_{inter}
- Directionnal derivative of $\widehat{x}^?$ are combination of $\widehat{A}_{1_{:,2}}$ and $\widehat{A}_{2_{:,1}}$

Conclusions

An ongoing work:

- Still don't know how to solve this new problem
- May be extends to more than 2 blocs
- Litterature review about bloc matrix resolution (ADMM)

References

A Griewank and Ph Toint. On the unconstrained optimization of partially separable functions. In M. J. D Powell, editor, *Nonlinear Optimization 1981*, pages 301–312. Academic press, 1982a. Publication editors : M.J.D. Powell.

A Griewank and Ph Toint. Partitioned variable metric updates for large structured optimization problems. 39:119–137, 1982b. doi: 10.1007/BF01399316.

A Conn, N Gould, M Lescrenier, and Ph Toint. Performance of a multifrontal scheme for partially separable optimization. In *Advances in Optimization and Numerical Analysis*, pages 79–96. Springer, 1994. doi: 10.1007/978-94-015-8330-5_6.

A Conn, N Gould, A Sartenaer, and Ph Toint. Convergence properties of minimization algorithms for convex constraints using a structured trust region. 6(4):1059–1086, 1996. doi: https://doi.org/10.1137/S1052623492236481.

C Bischof, A Bouaricha, P Khademi, and J Moré. Computing gradients in large-scale optimization using automatic differentiation. 9:185–194, 1997. doi: https://doi.org/10.1287/ijoc.9.2.185.

N Durand and J-M Alliot. Genetic crossover operator for partially separable functions. In *GP 1998, 3rd annual conference on Genetic Programming*, Madison, United States, 1998. URL `https://hal-enac.archives-ouvertes.fr/hal-00937718`.

M. Porcelli and Ph Toint. Exploiting problem structure in derivative free optimization, 2021.

S Kim, M Kojima, and Ph Toint. Recognizing underlying sparsity in optimization. 119(2):273–303, Jul 2009. ISSN 1436-4646. doi: 10.1007/s10107-008-0210-4. URL `https://doi.org/10.1007/s10107-008-0210-4`.