# *Multilevel optimization methods for the training of artificial neural networks*

E. Riccietti (IRIT-INP, Toulouse)

Joint work with: H. Calandra (Total)
S. Gratton (IRIT-INP, Toulouse)
X. Vasseur (ISAE-SUPAERO, Toulouse)

FGS 2019

Nice - 17 - 20 Septembre 2019

## Context

We consider large-scale nonlinear unconstrained optimization problems:

$$\min_{x} f(x)$$

# Context

We consider large-scale nonlinear unconstrained optimization problems:

$$\min_x f(x)$$

Classical iterative optimization methods:

$$f(x_k + s) \simeq T_2(x_k, s)$$

with $T_2(x_k, s)$ Taylor model of order 2. At each iteration we compute a step $s_k$ to update the iterate:

$$\min_s m_k(x_k, s) = T_2(x_k, s) + r(\lambda_k), \qquad \lambda_k > 0$$

$r(\lambda_k)$ regularization term.

# A classical example

- Adaptive Cubic Regularization method (ARC):

$$m_k(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{\lambda_k}{3} \|s\|^3$$

*Adaptive cubic regularisation methods for unconstrained optimization,* C. Cartis, N. Gould, Ph. Toint, 2009

# Extension to higher-order methods ($q > 2$)

Model of order $q$:

$$\min_s m_{q,k}(x_k, s) = T_q(x_k, s) + \frac{\lambda_k}{q+1}\|s\|^{q+1}, \qquad \lambda_k > 0.$$

$$T_q(x_k, s) = \sum_{i=1}^{q} \frac{1}{i!} \nabla^i f(x_k)(\overbrace{s, \ldots, s}^{i \text{ times}})$$

*Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models*, E. G. Birgin, J. L. Gardenghi, J. M. Martınez, S. A. Santos and Ph. L. Toint, 2017

# High order methods

Unifying framework for global convergence and worst-case complexity is presented.

☺ better complexity

☹ needs higher-order derivatives, model is expensive to minimize

*Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models*, E. G. Birgin, J. L. Gardenghi, J. M. Martınez, S. A. Santos and Ph. L. Toint, 2017

# Bottleneck: Subproblem solution

Solving

$$\min_s T_q(x_k, s) + \frac{\lambda_k}{q+1}\|s\|^{q+1}$$

represents greatest cost per iteration, which depends on the size of the problem.

$$\Downarrow$$

*Multilevel trust region method,* S. Gratton, A. Sartenaer, PH. Toint, 2008

## Hierarchy of problems

- $\{f^l(x^l)\}$, $x^l \in \mathcal{D}_l$
- $|\mathcal{D}_l| < |\mathcal{D}_{l+1}|$
- $f^l$ is cheaper to optimize compared to $f^{l+1}$

# Our contributions

📄 E. G. Birgin, J. L. Gardenghi, J. M. Martinez, S. A. Santos and Ph. L. Toint, 2017

- one level methods: non-scalable

📄 S. Gratton, A. Sartenaer, PH. Toint, 2008

- method for second order models

$$\Downarrow$$

We propose a family of scalable multilevel methods using high-order models.

# Outline

- Part I: multilevel extension of iterative high-order optimization methods
  - global convergence
  - worst-case complexity
  - local convergence rate

# Outline

- Part I: multilevel extension of iterative high-order optimization methods
  - global convergence
  - worst-case complexity
  - local convergence rate
- Part II: use of the multilevel methods for the training of artificial neural network
  - multilevel methods in the literature used just for problems with a geometrical structure

# Part I

1. Multilevel extension of iterative high-order optimization methods

# Multilevel setting

- At each level $l$, $x \in \mathbb{R}^{n_l}$. $l_{\max}$ finest level, 0 coarsest level.

$$\begin{array}{c|c|ccc} \text{level } l_{\max} & \mathbb{R}^{n} & x^{l_{\max}} & f^{l_{\max}} = f & \mu^{l_{\max}} = f \\ \vdots & & \vdots & \vdots & \vdots \\ \text{level } l+1 & \mathbb{R}^{n_{l+1}} & x^{l+1} & f^{l+1} & \mu^{l+1} \\ & R^{l+1} \Downarrow \quad \Uparrow P^{l+1} & & & \\ \text{level } l & \mathbb{R}^{n_l} & x^{l} & f^{l} & \mu^{l} \\ \vdots & & \vdots & \vdots & \vdots \\ \text{level } 0 & \mathbb{R}^{n_0} & x^{0} & f^{0} & \mu^{0} \end{array}$$

- $f^l$ represents $f$ on the coarse spaces (it is e.g. the discretization of $f$ on a coarse space)
- The functions $\mu^l$ are modifications of the $f^l$ to ensure inter-level coherence.
- $R^l = \alpha(P^l)^T$, for some $\alpha > 0$.

# One level strategy

At level $l = l_{\max}$, let $x_k^l$ be the current approximation. We look for a correction $s_k^l$ to define the new approximation $x_{k+1}^l = x_k^l + s_k^l$.

$x_k^l$

# One level strategy

At level $l = l_{\max}$, let $x_k^l$ be the current approximation. We look for a correction $s_k^l$ to define the new approximation $x_{k+1}^l = x_k^l + s_k^l$.

$$x_k^l \xrightarrow{\quad T_q^l \quad} x_{k+1}^l = x_k^l + s_k^l$$

# Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get $s_k^l$,
2. choose lower level model $\mu^{l-1}$:

$$\begin{tikzcd} x_k^l \arrow[r, "T_q^l"] & x_{k+1}^l = x_k^l + s_k^l \end{tikzcd}$$

# Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get $s_k^l$,
2. choose lower level model $\mu^{l-1}$:

# Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get $s_k^l$,
2. choose lower level model $\mu^{l-1}$:

$x_k^l$

# Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get $s_k^l$,
2. choose lower level model $\mu^{l-1}$:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] \\
R^l x_k^l := x_{0,k}^{l-1}
\end{tikzcd}$$

# Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get $s_k^l$,
2. choose lower level model $\mu^{l-1}$:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] & \\
R^l x_k^l := x_{0,k}^{l-1} \arrow[r, "\mu^{l-1}"] & x_{*,k}^{l-1}
\end{tikzcd}$$

# Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get $s_k^l$,
2. choose lower level model $\mu^{l-1}$:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] & x_{k+1}^l = x_k^l + s_k^l \\
R^l x_k^l := x_{0,k}^{l-1} \arrow[r, "\mu^{l-1}"] & x_{*,k}^{l-1} \arrow[u, "s_k^l = P^l(x_{*,k}^{l-1} - x_{0,k}^{l-1})"']
\end{tikzcd}$$

# Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get $s_k^l$,
2. choose lower level model $\mu^{l-1}$:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] & & x_{k+1}^l = x_k^l + s_k^l \\
R^l x_k^l := x_{0,k}^{l-1} \arrow[rr, "\mu^{l-1}"] & & x_{*,k}^{l-1} \arrow[u, "s_k^l = P^l(x_{*,k}^{l-1} - x_{0,k}^{l-1})"']
\end{tikzcd}$$

- The lower level model is cheaper to optimize.
- The procedure is recursive: more levels can be used.

# Coherence between levels, $q = 1$

Lower level model:

- Let $x_{0,k}^{l-1} = Rx_k^l$. Model with first order correction:

$$\mu_{1,k}^{l-1}(x_{0,k}^{l-1}, s^{l-1}) = f^{l-1}(x_{0,k}^{l-1}+s^{l-1})+(R^l \nabla f^l(x_k^l)-\nabla f^{l-1}(x_k^{l-1}))^T s^{l-1}$$

This ensures that

$$\nabla \mu_{1,k}^{l-1}(x_{0,k}^{l-1}) = R^l \nabla f^l(x_k^l)$$

→ first-order behaviours of $f^l$ and $\mu^{l-1}$ are coherent in a neighbourhood of the current approximation. If $s^l = P^l s^{l-1}$

$$\nabla f^l(x_k^l)^T s^l = \nabla f^l(x_k^l)^T P^l s^{l-1} = \nabla \mu_{1,k}^{l-1}(x_{0,k}^{l-1})^T s^{l-1}.$$

# Coherence between levels, $q = 2$

Lower level model: Let $x_{0,k}^{l-1} = Rx_k^l$. We define $\mu_{2,k}^{l-1}$ as

$$\mu_{2,k}^{l-1}(x_{0,k}^{l-1} + s^{l-1}) = f^{l-1}(x_{0,k}^{l-1} + s^{l-1}) + (R^l \nabla f^l(x_k^l) - \nabla f^{l-1}(x_k^{l-1}))^T s^{l-1}$$
$$+ \frac{1}{2}(s^{l-1})^T((R^l)^T \nabla f^l(x_k^l) P^l - \nabla^2 f^{l-1}(x_k^{l-1}))s^{l-1}$$

→ We can generalize this up to order $q$ to have the behaviours of $f^l$ and $\mu_{q,k}^{l-1}$ to be coherent up to order $q$ in a neighbourhood of the current approximation.

# Coherence up to order $q$

We define

$$\mu_{q,k}^{l-1}(x_{0,k}^{l-1}, s^{l-1}) = f^{l-1}(x_{0,k}^{l-1} + s^{l-1}) + \sum_{i=1}^{q} \frac{1}{i!}[\mathcal{R}(\nabla^i f^l(x_k)) - \nabla^i f^{l-1}(x_{0,k}^{l-1})]\underbrace{(s^{l-1}, \dots, s^{l-1})}_{i \text{ times}},$$

where $\mathcal{R}(\nabla^i f^l(x_k^l))$ is such that for all $i = 1, \dots, q$ and $s_1^{l-1}, \dots, s_i^{l-1} \in \mathbb{R}^{n_{l-1}}$

$$[\mathcal{R}(\nabla^i f^l(x_k^l))](s_1^{l-1}, \dots, s_i^{l-1}) := \nabla^i f^l(x_k^l, Ps_1^{l-1}, \dots, Ps_i^{l-1}),$$

where $\nabla^i f^l$ denotes the $i$-th order tensor of $f^l$.

# Theoretical results: Assumptions

## Assumption 1

Let us assume that for all $l$ the $q$-th derivative tensors of $f^l$ are Lipschitz continuous.

## Assumption 2

There exist strictly positive scalars $\kappa_{EB}, \rho > 0$ such that

$$\mathrm{dist}(x, \mathcal{X}) \leq \kappa_{EB} \|\nabla_x f(x)\|, \quad \forall x \in \mathcal{N}(\mathcal{X}, \rho),$$

where $\mathcal{X}$ is the set of second-order critical points of $f$, $\mathrm{dist}(x, \mathcal{X})$ denotes the distance of $x$ to $\mathcal{X}$ and $\mathcal{N}(\mathcal{X}, \rho) = \{x \mid \mathrm{dist}(x, \mathcal{X}) \leq \rho\}$.

*On the Quadratic Convergence of the Cubic Regularization Method under a Local Error Bound Condition*, Yue, M.C. and Zhou, Z. and So, A.M.C., 2018: generalized to higher-order methods

# Theoretical results: 1) global convergence

**Theorem**

*Let Assumption 1 hold. Then, the sequence of iterates generated by the algorithm converges globally to a first-order stationary point.*

# Theoretical results: 1) global convergence

**Theorem**

*Let Assumption 1 hold. Then, the sequence of iterates generated by the algorithm converges globally to a first-order stationary point.*

- E. G. Birgin, J. L. Gardenghi, J. M. Martınez, S. A. Santos and Ph. L. Toint, 2017: generalized to multilevel framework
- Gratton, Sartenaer, Toint, 2008: extended to higher-order models and simplified

# Theoretical results: 2) complexity

## Theorem

*Let Assumption 1 hold. Let $f_{low}$ be a lower bound on $f$. Then, the method requires at most*

$$K_3 \frac{(f(x_{k_1}) - f_{low})}{\epsilon^{\frac{q+1}{q}}} \left(1 + \frac{|\log \gamma_1|}{\log \gamma_3}\right) + \frac{1}{\log \gamma_3} \log\left(\frac{\lambda_{\max}}{\lambda_0}\right)$$

*iterations to achieve an iterate $x_k$ such that $\|\nabla f(x_k)\| \leq \epsilon$, where*

$$K_3 := \frac{q+1}{\eta_1 \lambda_{\min}} \max\{K_1^{1/q}, K_2^{1/q}\}.$$

E. G. Birgin, J. L. Gardenghi, J. M. Martınez, S. A. Santos and Ph. L. Toint, 2017: $k = O(\epsilon^{-\frac{q+1}{q}})$ Complexity of standard method is maintained

# Theoretical result: 3) local convergence

## Theorem

Let Assumptions 1 and 2 hold. Assume that $\mathcal{L}(f(x_k))$ is bounded for some $k \geq 0$ and that it exists an accumulation point $x^*$ such that $x^* \in \mathcal{X}$. Then, the whole sequence $\{x_k\}$ converges to $x^*$ and it exist strictly positive constants $c \in \mathbb{R}$ and $\bar{k} \in \mathbb{N}$ such that:

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^q} \leq c, \quad \forall k \geq \bar{k}.$$

# Theoretical result: 3) local convergence

## Theorem

Let Assumptions 1 and 2 hold. Assume that $\mathcal{L}(f(x_k))$ is bounded for some $k \geq 0$ and that it exists an accumulation point $x^*$ such that $x^* \in \mathcal{X}$. Then, the whole sequence $\{x_k\}$ converges to $x^*$ and it exist strictly positive constants $c \in \mathbb{R}$ and $\bar{k} \in \mathbb{N}$ such that:

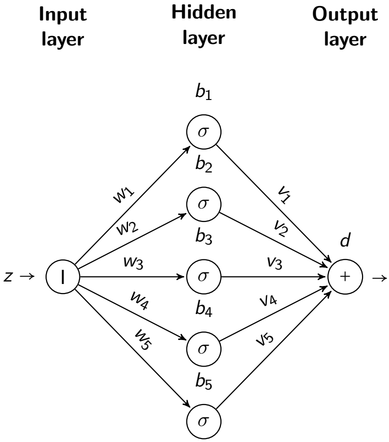
$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^q} \leq c, \quad \forall k \geq \bar{k}.$$

- E. G. Birgin, J. L. Gardenghi, J. M. Martınez, S. A. Santos and Ph. L. Toint, 2017: local convergence not proved
- Gratton, Sartenaer, Toint, 2008: local convergence not proved

## Part II

1. Use of the multilevel methods for the training of artificial neural networks

# Artificial neural networks



$$\hat{g}(p,z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$$

$z \in \mathbb{R}$,
$p = [v, w, b, d]^T$
$\hat{g}(p,z) \sim g(z)$

Activation funct. $\sigma$,
sigmoid: $\sigma(z) = \frac{e^z-1}{e^z+1}$,
tanh: $\sigma(z) = \frac{e^{2z}-1}{e^{2z}+1}$,
logit: $\sigma(z) = \frac{e^z}{e^z+1}$, softplus: $\sigma(z) = \log(e^z+1)$.

# Exploit multilevel method for training of ANNs

Training problem:

$$\min_p \mathcal{L}(p,z) = \mathcal{F}(\hat{g}(p,z) - g(z)), \qquad z \in \mathcal{T}$$

$$\hat{g}(p,z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$$

where $\mathcal{L}$ is the loss function, $\mathcal{T}$ training set.

# Exploit multilevel method for training of ANNs

Training problem:

$$\min_p \mathcal{L}(p,z) = \mathcal{F}(\hat{g}(p,z) - g(z)), \qquad z \in \mathcal{T}$$

$$\hat{g}(p,z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$$

where $\mathcal{L}$ is the loss function, $\mathcal{T}$ training set.

Large-scale problem: can we exploit multilevel methods for the training?

# Exploit multilevel method for training of ANNs

Training problem:

$$\min_p \mathcal{L}(p,z) = \mathcal{F}(\hat{g}(p,z) - g(z)), \qquad z \in \mathcal{T}$$

$$\hat{g}(p,z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$$

where $\mathcal{L}$ is the loss function, $\mathcal{T}$ training set.

**Large-scale problem: can we exploit multilevel methods for the training?**

- How to build the coarse problem? The variables to be optimized are the network's weights:
  NO evident geometrical structure to exploit!

# Exploit multilevel method for training of ANNs

Training problem:

$$\min_p \mathcal{L}(p,z) = \mathcal{F}(\hat{g}(p,z) - g(z)), \qquad z \in \mathcal{T}$$

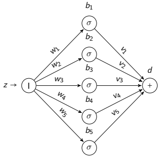
$$\hat{g}(p,z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$$

where $\mathcal{L}$ is the loss function, $\mathcal{T}$ training set.

**Large-scale problem: can we exploit multilevel methods for the training?**

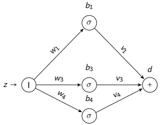
- How to build the coarse problem? The variables to be optimized are the network's weights:
  NO evident geometrical structure to exploit!
- The network possesses a purely algebraic structure: can we exploit it?

# Exploit multilevel method for training of ANNs



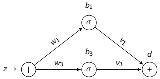
$$\mathcal{F}_1 : \mathbb{R}^{3r_1} \rightarrow \mathbb{R}$$
$$\hat{g}(p,z) = \textstyle\sum_{i \in I_1} v_i \sigma(w_i z + b_i) + d$$
$$|I_1| = r_1$$

$$R_1 \Downarrow P_1 \Uparrow$$



$$\mathcal{F}_2 : \mathbb{R}^{3r_2} \rightarrow \mathbb{R}$$
$$\hat{g}(p,z) = \textstyle\sum_{i \in I_2} v_i \sigma(w_i z + b_i) + d$$
$$I_2 \subset I_1,\ |I_2| = r_2 < r_1$$

$$R_2 \Downarrow P_2 \Uparrow$$



$$\mathcal{F}_3 : \mathbb{R}^{3r_3} \rightarrow \mathbb{R}$$
$$\hat{g}(p,z) = \textstyle\sum_{i \in I_3} v_i \sigma(w_i z + b_i) + d$$
$$I_3 \subset I_2,\ |I_3| = r_3 < r_2$$

# How do we select the hierarchy of variables?

Algebraic multigrid: C/F splitting

### Ruge and Stueben $C/F$ splitting for $Ax = b$

- Two variables $i, j$ are said to be *coupled* if $a_{i,j} \neq 0$.
- We say that a variable $i$ is strongly coupled to another variable $j$, if $-a_{i,j} \geq \epsilon \max_{a_{i,k}<0} |a_{i,k}|$ for a fixed $0 < \epsilon < 1$, usually $\epsilon = 0.25$.

### Prolongation-Restriction operators

$P = [I; \Delta]$, $R = P^T$.

# Which matrix should we use?

Assume to use a second-order model:

$$m(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{\lambda_k}{3} \|s\|^3$$
$$m(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T B_k s + \frac{\lambda_k}{2} \|s\|^2$$

At each iteration we have to solve a linear system of the form:

$$(B_k + \tilde{\lambda}_k I) s = -\nabla f(x_k), \quad \tilde{\lambda}_k > 0.$$

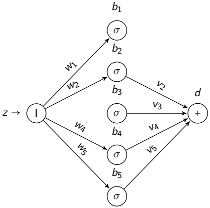
As in AMG for linear systems, we use information contained in matrix $B_k$.

# Which matrix should we use?

**Remark**

Variables are coupled!

$\{w_i, b_i, v_i\}$



We do not use the full matrix $B_k$ and we define $A$ as:

$$B_k = \begin{bmatrix} f_{v,v} & .. & .. \\ .. & f_{w,w} & .. \\ .. & .. & f_{b,b} \end{bmatrix} \rightarrow A = \frac{f_{v,v}}{\|f_{v,v}\|_\infty} + \frac{f_{w,w}}{\|f_{w,w}\|_\infty} + \frac{f_{b,b}}{\|f_{b,b}\|_\infty}$$

We define the coarse/fine splitting based on the auxiliary matrix $A$.

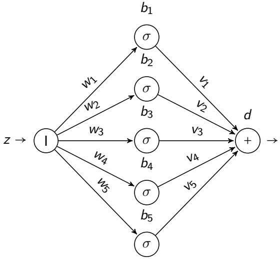
# Application: solution of PDEs

Approximate the solution $u(z)$ of a PDE:

$$D(z, u(z)) = g(z),\ z \in (a, b);$$
$$u(a) = A,\ u(b) = B.$$

We approximate the solution of the PDE with a neural network:

$$u(z) \sim \hat{u}(p, z), \quad p \in \mathbb{R}^n$$



$p = [v, w, b, d]^T$

$\hat{u}(p, z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$

# Application: solution of PDEs

## Advantages

- No need of discretization: we get an analytical expression of the solution, with good generalization properties (also for points outside the interval)
- Natural approach for solving nonlinear equations
- Alleviate the curse of dimensionality

Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations (2018).

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations (2019)

Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data (2018)

# Application: solution of PDEs

We select a training set $\mathcal{T}$ s.t. $|\mathcal{T}| = t$:

$$z = [z_1, \dots, z_t]^T, \quad a \le z_1 < \cdots < z_t \le b$$

We define

$$\mathcal{L}(p, z) = \frac{1}{2t}(\|D(z, \hat{u}(z)) - g(z)\|^2 + \lambda_p(\|\hat{u}(a) - A\|^2 + \|\hat{u}(b) - B\|^2))$$

for $\hat{u}(z) \in \mathbb{R}^t$.

Least-squares problem → multi-level Levenberg-Marquardt method

# Choice of the true solution

$$D(z, u(z)) = g(z),\ z \in (a, b);$$

- We choose $g$ to have true solution $u_T(z, \nu)$ depending on $\nu$

**Remark**

- As $\nu$ increases the function becomes more oscillatory and it is harder to approximate.
- The size of the problem increases with the number of nodes.
- $\mathcal{T}$: equispaced points in $(0, 1)$ with $h = \frac{1}{3\nu}$ (Shannon's criterion).

# Preliminary results: Poisson's equation 10 runs

| 1D | | $\nu = 20$ | $r = 2^9$ | | $\nu = 25$ | $r = 2^{10}$ |
|---|---|---|---|---|---|---|
| Solver | iter | RMSE | save | iter | RMSE | save |
| LM | 869 | 1.e-4 | | 1439 | 1.e-3 | |
| MLM | 507 | 1.e-4 | 1.1-2.6-4.3 | 1325 | 1.e-3 | 1.2-1.7-2.8 |

Table: 1D Poisson's equation, $u_T(z, \nu) = cos(\nu z)$, 10 runs

| 2D | | $\nu = 5$ | $r = 2^{10}$ | | $\nu = 6$ | $r = 2^{11}$ |
|---|---|---|---|---|---|---|
| Solver | iter | RMSE | save | iter | RMSE | save |
| LM | 633 | 1.e-3 | | 1213 | 1.e-3 | |
| MLM | 643 | 1.e-3 | 1.1-1.5-2.1 | 1016 | 1.e-3 | 1.2-1.9-2.4 |

Table: 2D Poisson's equation, $u_T(z, \nu) = cos(\nu z)$, 10 runs

save(min,average,max)=ratio between total number of flops required for matrix-vector products

# Helmholtz's and nonlinear equations, 10 runs

| | $\nu = 5$ | | $r = 2^{10}$ |
|---|---|---|---|
| Solver | iter | RMSE | save |
| LM | 1159 | 1.e-3 | |
| MLM | 1250 | 1.e-3 | 1.2-1.9-3.1 |

Table: Helmholtz's equations. $\Delta u(z) + \nu^2 u(z) = 0$ , $u_T(z,\nu) = sin(\nu z) + cos(\nu z)$

| | $\nu = 20$ | | $r = 2^9$ | $\nu = 1$ | | $r = 2^9$ |
|---|---|---|---|---|---|---|
| Method | iter | RMSE | save | iter | RMSE | save |
| LM | 950 | $10^{-5}$ | | 270 | $10^{-3}$ | |
| MLM | 1444 | $10^{-5}$ | 0.8-2.9-5.3 | 320 | $10^{-3}$ | 1.2-1.7-1.8 |

Table: Left: $\Delta u + \sin u = g_1$ (1D) $u_T(z,\nu) = 0.1\cos(\nu z)$. Right: $\Delta u + e^u = g_1$ (2D), $u_T(z,\nu) = \log\left(\frac{\nu}{z_1+z_2+10}\right)$

# Conclusions

- Theoretical contribution: We have presented a class of multilevel high-order methods for optimization and proved their global and local convergence and complexity.
- Practical contribution: We have got further insight on the methods proposing a AMG strategy to build coarse representations of the problem to use some methods in the family for the training of artificial neural networks.

# Perspectives

- Hessian-free method. Make it a competitive training method: the method needs to compute and store the Hessian matrix (for step computation and to build transfer operators): too expensive for large-scale problems.
- Extend the method to multilayer networks.

Thank you for your attention!

- *On the approximation of the solution of partial differential equations by artificial neural networks trained by a multilevel Levenberg-Marquardt method,* H. Calandra, S. Gratton, E. Riccietti X. Vasseur, submitted.
- *On high-order multilevel optimization strategies,* H. Calandra, S. Gratton, E. Riccietti X. Vasseur, submitted.
- *On the solution of systems of the form $A^T Ax = A^T b + c$,* H. Calandra, S. Gratton, E. Riccietti X. Vasseur, to be submitted.

If $q = 1$, the regularized model is defined as

$$f(x_k) + \nabla f(x_k) + \frac{\lambda_k}{2}\|s\|^2, \tag{1}$$

where in case of a least-squares problem $\nabla f(x_k) = J(x_k)^T F(x_k)$. For a positive definite matrix $M \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$, we can define the following norm:

$$\|x\|_M = x^T Mx.$$

If we define $M = \frac{B_k}{\lambda_k} + I$, then we have $\frac{\lambda_k}{2}\|s\|_M^2 = \frac{1}{2}s^T B_k s + \frac{\lambda_k}{2}\|s\|^2$, so that the model

$$m_k(x_k, s) = f(x_k) + \nabla f(x_k) + \frac{\lambda_k}{2}\|s\|_M^2,$$

corresponds to $q = 1$, just with a different norm for the regularization term.

# Tensor of order 3

## Definition

Let $T \in \mathbb{R}^{n^3}$, and $u, v, w \in \mathbb{R}^n$. Then $T(u, v, w) \in \mathbb{R}$, $T(v, w) \in \mathbb{R}^n$

$$T(u, v, w) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} T(i, j, k) u(i) v(j) w(k),$$

$$T(v, w)(i) = \sum_{j=1}^{n} \sum_{k=1}^{n} T(i, j, k) v(j) w(k), \quad i = 1, \ldots, n.$$

# Tensor of order $i$

## Definition

Let $i \in \mathbb{N}$ and $T \in \mathbb{R}^{n^i}$, and $u \in \mathbb{R}^n$. Then $T(\underbrace{u, \ldots, u}_{i \textit{ times}}) \in \mathbb{R}$,

$T(\underbrace{u, \ldots, u}_{i-1 \text{ times}}) \in \mathbb{R}^n$ and

$$T(\underbrace{u, \ldots, u}_{i \text{ times}}) = \sum_{j_1=1}^{n} \cdots \sum_{j_i=1}^{n} T(j_1, \ldots, j_i) u(j_1) \ldots u(j_i),$$

$$T(\underbrace{u, \ldots, u}_{i-1 \text{ times}})(j_1) = \sum_{j_2=1}^{n} \cdots \sum_{j_i=1}^{n} T(j_1, \ldots, j_i) u(j_2), \ldots u(j_i), \quad j_1 = 1, \ldots, n.$$

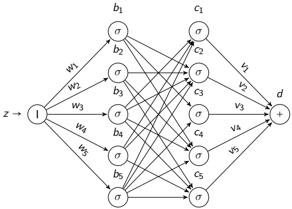
# When to use the lower level model?

The lower level model is not always useful, we can use it if

- if $\|\nabla\mu_{q,k}^{l-1}(x_{0,k}^{l-1})\| = \|R^l \nabla f^l(x_k^l)\| \geq \kappa \|\nabla f^l(x_k^l)\|$, $\kappa > 0$,
- if $\|R \nabla f^l(x_k^l)\| > \epsilon^l$

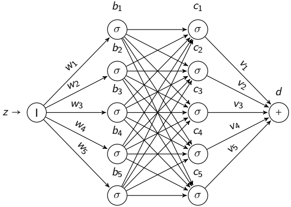
# Future work 1: Extend the method to multilayer networks.

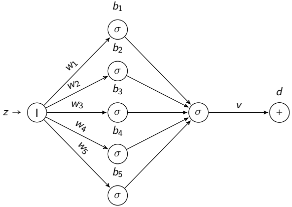
- Extend the method as it is: use a sparse network.



# Future work 1: Extend the method to multilayer networks.

- Extend the method as it is: use a sparse network.
- Change strategy to build coarse problems: compress variables in a layer to exploit the structure of the multilayer network.





**Thank you for your attention!**

For more details:

- On high-order multilevel optimization strategies and their application to the training of artificial neural networks

# Prolongation operator

$$x_i^h = (Px^H)_i = \begin{cases} x_i^H & \text{if } \ i \in C, \\ \sum_{k \in P_i} \delta_{i,k} x_k^H & \text{if } \ i \in F, \end{cases}$$

with

$$\delta_{i,k} = \begin{cases} -\alpha_i a_{i,k} / a_{i,i} & \text{if } \ k \in P_i^-, \\ -\beta_i a_{i,k} / a_{i,i} & \text{if } \ k \in P_i^+, \end{cases} \qquad \alpha_i = \frac{\sum_{j \in N_i} a_{i,j}^-}{\sum_{k \in P_i} a_{i,k}^-}, \qquad \beta_i = \frac{\sum_{j \in N_i} a_{i,j}^+}{\sum_{k \in P_i} a_{i,k}^+},$$

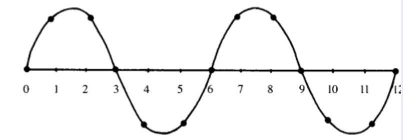
where $a_{i,j}^+ = \max\{a_{i,j}, 0\}$, $a_{i,j}^- = \min\{a_{i,j}, 0\}$, $N_i$ is the set of variables connected to $i$ (i.e. all $j$ such that $a_{i,j} \neq 0$), $P_i$ the set of coarse variables strongly connected to $i$, which is partitioned in $P_i^-$ (negative couplings) and $P_i^+$ (positive couplings). The interpolation operator, assuming to have regrouped and ordered the variables to have all those corresponding to indexes in $C$ at the beginning, is then defined as $P = [I; \Delta]$ where $I$ is the identity matrix of size $|C|$ and $\Delta$ is the matrix such that $\Delta_{i,j} = \delta_{i,j}$.

# Classical multigrid methods

- Consider a linear elliptic PDE: $D(z,u(z)) = f(z)\ z \in \Omega$ + b.c.
- Discretize on grid $h$. Get a large-scale linear system $A_h x_h = b_h$.

Consider the discretization of the same PDE problem on a coarser grid: $A_H x_H = b_H$, $H > h$.

- Relaxation methods fails to eliminate smooth components of the error efficiently.
- Smooth components projected on a coarser grid appear more oscillatory.



$k$ = 4 wave on $n$ = 12 grid

# Coarse problem construction

Define transfer grid operators: $P$ prolongation and $R$ restriction to project vectors from a grid to another: $x_H = Rx_h$, $x_h = Px_H$, such that $R = \alpha P^T$.

## Geometry exploitation

The geometrical structure of the problem is exploited to build $R$ and $P$.

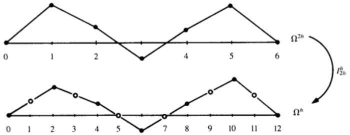
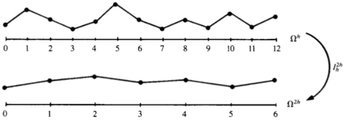


Figure 3.2: *Interpolation of a vector on coarse grid $\Omega^{2h}$ to fine grid $\Omega^h$.*


0
1
2
3
4
5
6
7
8
9
10
11
12
$\Omega^{h}$
$I_{h}^{2h}$
$\Omega^{2h}$
0
1
2
3
4
5
6