Piecewise linear bounding of univariate nonlinear functions and resulting MILP-based solution methods

Sandra Ulrich NGUEVEU

Institut National Polytechnique de Toulouse - Laboratoire LAAS-CNRS
ngueveu@laas.fr

OR@DII Seminar - Brescia - 23/10/2019
Outlook

1. Motivation and key ideas
2. Continuous PWL approximation and absolute tolerance
3. nnc PWL approximation/bounding and relative tolerance
4. Corridors fitting and generalization to classes of errors
5. Conclusion
Energy in hybrid electric vehicles

Electric propulsion motor powered by:
- onboard generator: e.g. hydrogen fuel cell (FC)
- reversible source: e.g. supercapacitator (SE)

Energy sources characteristics: **power limits** (kW), **efficiency** (%), **capacity** (kWs) ...

Find at each instant the **optimal power split** between the energy sources to **minimize the total fuel consumption**.

Power from FC + Power from/to SE = Total power provided
Mathematical model

\[
\min \sum_{i=1}^{n} f^{FC}(x_i)
\]  

(1)

s.t. Power demand satisfaction

\[
x_i + y_i - z_i \geq P_i, \quad \forall i \in \mathbb{I}
\]  

(2)

Final SE energy level higher or equal to initial energy level

\[
\sum_{i=1}^{n} f^{SE^+}(y_i) - f^{SE^-}(z_i) \leq 0
\]  

(3)

SE energy level within bounds

\[
E_{0}^{SE} - E_{max} \leq \sum_{k=1}^{i} f^{SE^+}(y_k) - f^{SE^-}(z_k) \leq E_{0}^{SE} - E_{min}, \quad \forall i \in \mathbb{I}
\]  

(4)

Variables domains

\[
x_i \in \{0\} \cup [P_{min}^{FC}, P_{max}^{FC}], y_i \in [0, P_{max}^{SE}], z_i \in [0, P_{min}^{SE}], \quad \forall i \in \mathbb{I}
\]  

(5)
Water pumping and desalination process

Electrical model
- $V_m, I_m$: electrical tension, current
- $T_m$: motor electromag. torque
- $\Omega$: rotation speed
- $k_\Phi$: torque equivalent coefficient
- $r$: stator resistance

Electric motor equations (inertia neglected):
\begin{align}
V_m &= rI_m + k_\Phi \Omega \\
T_m &= \Phi_m I_m
\end{align}

(6) \hspace{1cm} (7)

Electrical power needed: $P_e = V_m I_m$.

Pressure drop in the pipe
- $\Delta \text{Pipe}, \rho$: pressure drop, water density
- $h$: height of water pumping

Static+Dynamic pressure
\[ \Delta \text{Pipe} = k q^2 + \rho g h \]

(8)

Mechanical-Hydraulic conv.
- $P_p$: output pressure
- $q$: debit of water
- $a, b$: non linear girator coefs
- $c$: hydraulic friction
- $p_0$: suction pressure
- $s_p + s_m$: mechanical losses

Static equations of the motor-pump (mechanical inertia neglected):
\begin{align}
P_p &= (a\Omega + bq)\Omega - (cq^2 + p_0) \\
T_m &= (a\Omega + bq)q + (s_m + s_p)\Omega
\end{align}

(9) \hspace{1cm} (10)
Efficiency function of pump 2 + RO module

Subsystem pump 2 + Reverse Osmosis module is modeled with

\[ P_e = r \cdot K(q, h) + ((s_m + s_p) \cdot \Omega(q, h) + (q + F(q)/R_{Me}) \cdot M(q, h)) \cdot \Omega(q, h) \]

where

\[
\begin{align*}
F(q) &= (R_{Mod} + R_{Valve}) \cdot q^2 \\
G(q) &= (b \cdot (q + F(q)/R_{Me})) \\
M(q, h) &= a \cdot \Omega(q, h) + G(q) \\
\Omega(q, h) &= -G(q) + \sqrt{G(q)^2 - 4a \cdot ((\rho_0 + \rho g \cdot (h - l_{out})) + (k + c) \cdot ((q + F(q)/R_{Me})^2 + F(q)))} \\
K(q, h) &= (((s_m + s_p) \cdot \Omega(q, h) + (q + F(q)/R_{Me}) \cdot (a \cdot \Omega(q, h) + G(q)))) / k_0^2
\end{align*}
\]
Multicommodity network design problem with congestion

Minimize Total cost = design cost + routing cost + capacity augmentation cost + congestion cost

s.t.

- Flow conservation
- Maximum capacity (with/without upgrade)
- all commodities get to destination

Paraskevopoulos, Sinan Gürel and Bektas, 2016
Classical MINLP solution methods

MILP-based solution methods on similar problems
Camponogara et al. 2011; Borghetti et al., 2008

- approximate with piecewise linear functions

+ (more) tractable problems
- try and error approach: No guarantees on the solution quality or iterative process with an undefined number of iterations
- global optimality cannot be guaranteed

Generic MINLP solution methods / Hybrid algorithms and frameworks
Grossmann 2002 / Bonami et al., 2008

+ global optimality guaranteed if carried out to completion
- only for small/medium instances
Classical MINLP solution methods

MILP-based solution methods on similar problems
Camponogara et al. 2011; Borghetti et al., 2008

- approximate with piecewise linear functions

+ (more) tractable problems
- try and error approach: No guarantees on the solution quality or iterative process with an undefined number of iterations
- global optimality cannot be guaranteed

Generic MINLP solution methods / Hybrid algorithms and frameworks
Grossmann 2002 / Bonami et al., 2008

+ global optimality guaranteed if carried out to completion
- only for small/medium instances
Classical MINLP solution methods

MILP-based solution methods on similar problems
Camponogara et al. 2011; Borghetti et al., 2008

- approximate with piecewise linear functions

+ (more) tractable problems
  - try and error approach: No guarantees on the solution quality or iterative process with an undefined number of iterations
  - global optimality cannot be guaranteed

Generic MINLP solution methods / Hybrid algorithms and frameworks
Grossmann 2002 / Bonami et al., 2008

+ global optimality guaranteed if carried out to completion
  - only for small/medium instances
MILP-based solution methods for MINLPs

- Approximating nonlinear functions with PWL functions
- Modeling the PWL functions in a MILP
- Solving MILPs containing PWL functions

satisfactory solution?

Yes → DONE

No → Repeat steps
Approximating nonlinear functions with PWL functions

**Sampling**

Given a number of breakpoints on the curve, find the PWL functions which minimizes an error metric.

... D’Ambrosio et al 2010 ... D’Ambrosio et al 2015 ...

**Fitting a discrete data set**

Applications: ECG, pattern recognition, data reduction, ...

Bellman and Roth 1969 ... Tomek 1974 ...
O'Rourke 1981 ... Toriello and Vielma 2012 ... Rebennack and Krasko 2019
From an iterative to a non-iterative solution method?

Approximating nonlinear functions with PWL functions

Modeling the PWL functions in a MILP

Solving MILPs containing PWL functions

satisfactory solution?

Yes

DONE

No

Bound each nonlinear function with 2 PWL functions

Modeling the PWL functions in a MILP

Solving MILPs containing PWL functions

DONE
New two-step solution scheme: Ngueveu et al., 2016, 2019

PGMO projects OREM (2014-2016), OPAL (2016-2018)

Step 1: Piecewise linear **bounding** of the nonlinear energy transfer/efficiency functions

(c) Linear approximation  
(d) Piecewise bounding

Step 2: Reformulation of the problem into two mixed integer problems (MILP)

- solve with a MILP solver
- or design a dedicated solution method (only one needed)
Resulting PWL approximation/bounding Problem

Given a function \( f : \mathbb{R} \rightarrow \mathbb{R} \) and a tolerance bound, find a PWL function that:

- Respects the imposed bounded error
- Minimizes the number of pieces of the PWL function

Benefits

- **predetermined error criteria** \( \rightarrow \) **non-iterative** approach

  - Bound each nonlinear function with 2 PWL functions
  - Modeling the PWL functions in a MILP
  - Solving MILPs containing PWL functions
  - **DONE**

- **Smallest MILP possible!**
Why is the Problem Hard?

Bounded tolerance constraints $\rightarrow$ semi-infinite programming (SIP)

Maximizing the length of the projection on x-axis of each segment is not optimal
### Approximation of continuous functions with PWL functions

Minimize number of linear pieces for a bounded error or distance metric

**Rosen and Pardalos, 1986.** Global minimization of large-scale constrained concave quadratic problems by separable programming. *Math. Prog*

**Frenzen, Sasao and Butler, 2010.** On the number of segments needed in a piecewise linear approximation. *J. of Comp. and Applied Math.*

**Geibler, Martin, Morsi, Schewe, 2012.** Using piecewise linear functions for solving MINLPs. *The IMA Volumes in Mathematics and its applications*

**Rebennack and Kallrath, 2015.** Continuous piecewise linear $\delta$-approximations for univariate functions: computing minimal breakpoint system. *J. Optim. Theory Appl*

**Ngueveu, 2019.** Piecewise linear bounding of univariate nonlinear functions and resulting mixed integer linear programming-based solution methods *EJOR*

**Rebennack and Krasko, 2019.** Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing*
1 Motivation and key ideas

2 Continuous PWL approximation and absolute tolerance

3 nnc PWL approximation/bounding and relative tolerance

4 Corridors fitting and generalization to classes of errors

5 Conclusion
Continuous PWL approximation of univariate functions

Optimal Breakpoint System using a Continuum approach for \( x \) [RK2015]

Decision variables

- \( x_b \in [X_-, X_+] \) : breakpoint value
- \( s_b \in [-\delta, +\delta] \) : deviation on bpt b
- \( \chi_b \in [0, 1] = 1 \) iff bpt b is used
- \( y_b \geq \frac{1}{M} : = x_b - x_{b-1} \) if \( x_b - x_{b-1} > 0 \) and \( = |X_-, X_+| \) otherwise
- \( \xi_{bx}^x \in [0, 1] : = 1 \) iff \( x \in [x_{b-1}, x_b] \)
- \( l_b(x) \in \mathbb{R} : = l(x) \) if \( x \in [x_{b-1}, x_b] \)

Semi-infinite programming model

Obj = minimize number of active breakpoints

s.t. (c1) Order active breakpoints
(c2) Link \( x_b \) and \( \chi_b \)
(c3) Compute \( y_b \) and \( l_b(x) \)
(c4) Compute \( \xi_{bx}^x \)
(c5) Compute \( l(x) \)
(c6) Ensure the \( \delta \)-approximation : \( |l(x) - f(x)| \leq \delta, \forall x \in D \)
Continuous PWL approximation of univariate functions

Solution 1: Semi-infinite programming models (large and difficult)

Iterative solution method

- enforce the $\delta$ gap constraint on discrete points $|\|$
  - Rebennack and Kallrath 2015 $\Rightarrow$ solve a MINLP
  - Rebennack and Krasko 2019 $\Rightarrow$ solve a MILP (extension of Toriello and Vielma 2012)

- compute the real maximum error
  - Rebennack and Kallrath 2015, 2019 $\Rightarrow$ solve an NLP
Continuous PWL approximation of univariate functions

Solution 2: Greedy heuristics computing $x_b$ given $x_{b-1}, s_{b-1}$ and $\delta$

- Max length of interval $[x_{b-1}, x_b]$ (projection on x-axis): not optimal

- Solve the NLP problem of max $x_b$ or try discrete values of $x_b$ and $s_b$
1. Motivation and key ideas

2. Continuous PWL approximation and absolute tolerance

3. nnc PWL approximation/bounding and relative tolerance

4. Corridors fitting and generalization to classes of errors

5. Conclusion
Non necessarily continuous (nnc) PWL approximation

Proposition

Any optimal continuous $\delta$-PWLA with $n^*$ pieces can be converted into a non-necessarily continuous $\delta$-PWLA with $n \leq n^*$ pieces where the projection of the first piece on interval $D$ is of maximal length.
Non necessarily continuous (nnc) δ-PWLA

Getting rid of the continuity

Theorem

∀ continuous function \( f : \mathbb{D} = [X_-, X_+] \to \mathbb{R} \) and any scalar \( \delta \in \mathbb{R}^+ \), there exists an optimal nnc δ-PWLA \( g \) defined by

\[
G = \bigcup_{i=1}^{n_g} ([a_i, b_i], [x_{i,\min}, x_{i,\max}]) \text{ such that each line-segment } i \text{ has a maximal length projection on the interval } [x_{i,\min}, X_+].
\]

The greedy algorithm becomes optimal.
Non necessarily continuous (nn) $\delta$-PWLA

Getting rid of the continuity

**Theorem**

\[
\forall \text{ continuous function } f : \mathbb{D} = [X_{-}, X_{+}] \to \mathbb{R} \text{ and any scalar } \delta \in \mathbb{R}^+, \text{ there exists an optimal nn} \ \delta\text{-PWLA } g \text{ defined by } \\
G = \bigcup_{i=1}^{n_g} ([a_i, b_i], [x_{i}^{\text{min}}, x_{i}^{\text{max}}]) \text{ such that each line-segment } i \text{ has a maximal length projection on the interval } [x_{i}^{\text{min}}, X_{+}].
\]

The greedy algorithm becomes optimal
Non necessarily continuous (nnc) $\delta$-PWLA

Getting rid of the continuity

**Theorem**

\[ \forall \text{ continuous function } f : \mathbb{D} = [X_-, X_+] \to \mathbb{R} \text{ and any scalar } \delta \in \mathbb{R}^+ \text{, there exists an optimal nnc } \delta \text{-PWLA } g \text{ defined by } G = \bigcup_{i=1}^{n_g} ([a_i, b_i], [x_i^{\text{min}}, x_i^{\text{max}}]) \text{ such that each line-segment } i \text{ has a maximal length projection on the interval } [x_i^{\text{min}}, X_+]. \]

The greedy algorithm becomes optimal

\[ \frac{n^* + 1}{2} \leq n_g \leq n^* \]
Better $\delta$-PWL approximation algorithms

From $\delta$-approx to $\frac{\delta}{2}$-under(-over)estimator : shift by $\frac{\delta}{2}$

Convex or concave function

optimal 2-steps dichotomic search : No NLP or MINLP solved!

$\max \tilde{x}$

$\max x_{max_i}$

Piecewise convex or concave function with $p$ pieces

approx/bound each convex or concave piece separately

$n_g^* \leq n_g \leq n_g^* + p$
Numerical comparisons
Computational evaluation on continuous functions
Computational evaluation on continuous functions
Computational evaluation on continuous functions

\[ f(x) = 2x^2 + x^3 \]
\[ f(x) = e^{-x} \sin(x) \]
\[ f(x) = e^{-100(x-2)^2} \]
\[ f(x) = 1.03e^{-100(x-1.2)^2} + e^{-100(x-2)^2} \]

<table>
<thead>
<tr>
<th>( f )</th>
<th>( \delta )</th>
<th>continuous approx. [\text{[RK2015]}]</th>
<th>nnc approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>( \delta )</td>
<td>( n^* )</td>
<td>( \text{cpu} )</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>0.01</td>
<td>25</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>0.005</td>
<td>35</td>
</tr>
<tr>
<td>II</td>
<td>1</td>
<td>0.01</td>
<td>9</td>
</tr>
<tr>
<td>II</td>
<td>1</td>
<td>0.005</td>
<td>13</td>
</tr>
<tr>
<td>III</td>
<td>2</td>
<td>0.01</td>
<td>13</td>
</tr>
<tr>
<td>III</td>
<td>2</td>
<td>0.005</td>
<td>17</td>
</tr>
<tr>
<td>IV</td>
<td>2</td>
<td>0.01</td>
<td>9</td>
</tr>
<tr>
<td>IV</td>
<td>2</td>
<td>0.005</td>
<td>13</td>
</tr>
<tr>
<td>V</td>
<td>4</td>
<td>0.01</td>
<td>9</td>
</tr>
<tr>
<td>V</td>
<td>0.005</td>
<td>12</td>
<td>Few min</td>
</tr>
<tr>
<td>( f )</td>
<td>( \delta )</td>
<td>( p )</td>
<td>( n^* )</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>VI</td>
<td>0.05</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>15</td>
</tr>
<tr>
<td>VII</td>
<td>0.05</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>4</td>
</tr>
<tr>
<td>VIII</td>
<td>0.05</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>4</td>
</tr>
<tr>
<td>IX</td>
<td>0.05</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>7</td>
</tr>
</tbody>
</table>

**Motivation and key ideas**

- **continuous PWL approximation**
- **nnc PWL fct**

**Fitting corridors**

**Conclusion**

- **exact heuristic**
- **nnc approximation**

- \([\text{Ng2019}]\)

**Table:**

<table>
<thead>
<tr>
<th>( f )</th>
<th>( \delta )</th>
<th>( p )</th>
<th>( n^* )</th>
<th>( n^- )</th>
<th>( n^+ )</th>
<th>( \text{cpu} )</th>
<th>( n^* )</th>
<th>( \text{cpu} )</th>
<th>( n^- )</th>
<th>( n^+ )</th>
<th>( \text{cpu} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI</td>
<td>0.05</td>
<td>2</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>15</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>15</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td>0.05</td>
<td>3</td>
<td>4</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>4</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>4</td>
<td>61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIII</td>
<td>0.05</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>4</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>4</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>0.05</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>7</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>7</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**New Table:**

<table>
<thead>
<tr>
<th>( f )</th>
<th>( \delta )</th>
<th>( p )</th>
<th>( n^* )</th>
<th>( n^- )</th>
<th>( n^+ )</th>
<th>( \text{cpu} )</th>
<th>( n^* )</th>
<th>( \text{cpu} )</th>
<th>( n^- )</th>
<th>( n^+ )</th>
<th>( \text{cpu} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI</td>
<td>0.05</td>
<td>2</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>15</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>15</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td>0.05</td>
<td>3</td>
<td>4</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>4</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>4</td>
<td>61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIII</td>
<td>0.05</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>4</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>4</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>0.05</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01</td>
<td>7</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.005</td>
<td>7</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sandra Ulrich Ngueveu**

**Seminar AO DII 2019: PWLB of univariate non linear function**

23 / 41
<table>
<thead>
<tr>
<th>$f$</th>
<th>$\delta$</th>
<th>[\text{continuous approximation [RK2019]}]</th>
<th>[\text{nnc approximation [Ng2019]}]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(n_*)</td>
<td>(n_-)</td>
</tr>
<tr>
<td>VI</td>
<td>0.05</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td>0.05</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>VIII</td>
<td>0.05</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>0.05</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>
Weaknesses/Limitations of PWL-based solution methods

- dependent on the instance: e.g. dependent on time horizon
- dependent on the solution: e.g. solution cost
- difficult to choose a relevant value of $\delta$ for a new instance or worse a new problem

Energy optimization in hybrid electric vehicle

\[
\min_{i=1}^{n} f_{FC}(x_i)
\]  \hspace{1cm} (11)

s.t.
(c1) Power demand satisfaction
(c2) Final SE energy level higher or equal to initial energy level
(c3) SE energy level within bounds
(c5) Variables domains
Weaknesses/Limitations of PWL-based solution methods

Minimize the total energy cost

\[(\text{CF})\min\quad F = \sum_{t \in T} f\left(\sum_{i \in A} b_i x_{it}\right)\]  

(12)

s.t. Satisfaction of the demand for each activity

\[\sum_{t \in T} a_{it} x_{it} \geq p_i, \quad \forall i \in A\]  

(13)

Validity domain

\[x_{it} \in \{0, 1\}, \quad \forall i \in A, t \in T\]  

(14)
Use relative $\epsilon$-tolerance

**Principle**
For a function $f : \mathbb{D} \rightarrow \mathbb{R}^*$ and a tolerance value $\epsilon \in [0, 1]$, identify two piecewise linear functions $(\underline{f}^\epsilon, \overline{f}^\epsilon)$ that verify:

\[
\underline{f}^\epsilon(x) \leq f(x) \leq \overline{f}^\epsilon(x), \quad \forall x \in \mathbb{D}
\]

\[
|f(x) - \underline{f}^\epsilon(x)| \leq \epsilon|f(x)|, \quad \forall x \in \mathbb{D}
\]

\[
|\overline{f}^\epsilon(x) - f(x)| \leq \epsilon|f(x)|, \quad \forall x \in \mathbb{D}
\]

**Purpose**
- Apply on nonlinear term **BEFORE** insertion in the model
  - univariate function that can be convex or concave, easier to bound
  - minimize number of pieces ($\neq$ spatial or interval branch-and-bound)
- Two MILP ($\overline{\text{MILP}}$ and $\underline{\text{MILP}}$) are obtained
- Guarantees on the quality of the resulting lower and upper bounds
- Upper and lower bounding PWL functions $\overline{f}^\epsilon(x)$ and $\underline{f}^\epsilon(x)$ may not share the same breakpoints and number of pieces (**no shift !**)
Numerical comparisons

Comparison on the energy optimization problem for hybrid electric vehicles

- upper bounds
- lower bounds
- absolute vs relative tolerance
### Comparison of upper bounds

<table>
<thead>
<tr>
<th>Instances</th>
<th>(\epsilon)-PWLB+MILP [Ng2019]</th>
<th>Effpts+MILP [GCL2013]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\bar{n}^e) gap orig cpu gap recomp</td>
<td>gap cpu</td>
</tr>
<tr>
<td>(R) 6</td>
<td>(\bar{n}^e)</td>
<td>gap original cpu gap recomputation</td>
</tr>
<tr>
<td>6</td>
<td>0.20 %</td>
<td>16 s</td>
</tr>
<tr>
<td>56</td>
<td>0.01 %</td>
<td>58 s</td>
</tr>
<tr>
<td>(A1) 6</td>
<td>0.26 %</td>
<td>31 s</td>
</tr>
<tr>
<td>10</td>
<td>0.01 %</td>
<td>78 s</td>
</tr>
<tr>
<td>82</td>
<td>0.52 %</td>
<td>96 s</td>
</tr>
<tr>
<td>14</td>
<td>0.01 %</td>
<td>627 s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instances</th>
<th>antigen</th>
<th>baron</th>
<th>couenne</th>
<th>lindoglobal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>optCR</td>
<td>gap</td>
<td>gap</td>
<td>gap</td>
</tr>
<tr>
<td>(R) 6</td>
<td>1 %</td>
<td>13.86 %</td>
<td>9.64 %</td>
<td>20.97 % (4)</td>
</tr>
<tr>
<td>0.01 %</td>
<td>3.78 %</td>
<td>9.69 %</td>
<td>5.40 % (4)</td>
<td>0.01 % (5)</td>
</tr>
<tr>
<td>(A1) 6</td>
<td>1 %</td>
<td>1.06 %</td>
<td>8.62 %</td>
<td>3.44 % (5)</td>
</tr>
<tr>
<td>0.01 %</td>
<td>1.71 %</td>
<td>10.15 %</td>
<td>3.44 % (5)</td>
<td>0.37 % (5)</td>
</tr>
<tr>
<td>(A2) 6</td>
<td>1 %</td>
<td>0.96 %</td>
<td>14.74 %</td>
<td>20.11 % (2)</td>
</tr>
<tr>
<td>0.01 %</td>
<td>0.94 %</td>
<td>14.73 %</td>
<td>20.11 % (2)</td>
<td>0.00 % (5)</td>
</tr>
</tbody>
</table>
## Comparison of lower bounds

<table>
<thead>
<tr>
<th>Instances</th>
<th>old lower bound [NCMG2019]</th>
<th>$\epsilon$-PWLB+MILP [Ng2019]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set # ratio cpu</td>
<td>$\epsilon$ ratio cpu</td>
</tr>
<tr>
<td>(R)</td>
<td>6 97.81 % 2 s</td>
<td>1 % 99.18 % 20 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01 % 99.99 % 58 s</td>
</tr>
<tr>
<td>(A1)</td>
<td>6 97.36 % 2 s</td>
<td>1 % 99.25 % 29 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01 % 99.99 % 216 s</td>
</tr>
<tr>
<td>(A2)</td>
<td>6 78.50 % 2 s</td>
<td>1 % 99.50 % 26 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.01 % 99.99 % 405 s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instances</th>
<th>antigone</th>
<th>baron</th>
<th>couenne</th>
<th>lindoglobal</th>
</tr>
</thead>
<tbody>
<tr>
<td>class #</td>
<td>optCR</td>
<td>ratio</td>
<td>ratio</td>
<td>ratio</td>
</tr>
<tr>
<td>(R) 6</td>
<td>1 %</td>
<td>24.66 %</td>
<td>- % (6)</td>
<td>- % (6)</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>21.32 %</td>
<td>- % (6)</td>
<td>- % (6)</td>
</tr>
<tr>
<td>(A1) 6</td>
<td>1 %</td>
<td>49.41 %</td>
<td>37.89 % (0)</td>
<td>33.03 % (3)</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>49.60 %</td>
<td>37.72 % (0)</td>
<td>33.07 % (3)</td>
</tr>
<tr>
<td>(A2) 6</td>
<td>1 %</td>
<td>53.21 %</td>
<td>- % (6)</td>
<td>- % (6)</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>53.21 %</td>
<td>- % (6)</td>
<td>- % (6)</td>
</tr>
</tbody>
</table>
### Comparison of relative vs absolute tolerance

<table>
<thead>
<tr>
<th>Class (R)</th>
<th>using relative tol</th>
<th>using absolute tol</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance</td>
<td>$\epsilon$</td>
<td>$n^\epsilon$</td>
<td>UB</td>
</tr>
<tr>
<td>S_40</td>
<td>1 %</td>
<td>6</td>
<td>454.3</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>56</td>
<td>453.4</td>
</tr>
<tr>
<td>I_561</td>
<td>1 %</td>
<td>6</td>
<td>8756.6</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>56</td>
<td>8741.0</td>
</tr>
<tr>
<td>H_734</td>
<td>1 %</td>
<td>6</td>
<td>18626.0</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>56</td>
<td>18569.0</td>
</tr>
<tr>
<td>U_811</td>
<td>1 %</td>
<td>6</td>
<td>2613.5</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>56</td>
<td>2607.7</td>
</tr>
<tr>
<td>N_1200</td>
<td>1 %</td>
<td>6</td>
<td>23137.7</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>56</td>
<td>23114.9</td>
</tr>
<tr>
<td>E_1400</td>
<td>1 %</td>
<td>6</td>
<td>27088.9</td>
</tr>
<tr>
<td></td>
<td>0.01 %</td>
<td>56</td>
<td>27065.9</td>
</tr>
</tbody>
</table>

58% to 355% more linear pieces with absolute vs relative tolerance
Motivation and key ideas

Continuous PWL approximation and absolute tolerance

nnc PWL approximation/bounding and relative tolerance

Corridors fitting and generalization to classes of errors

Conclusion
PWL bounding with relative $\epsilon$-tolerance

**Strength**
- efficiency for problems with a nonlinear function per data set
- various fields and domains of application

**To be improved**
- access/ease-of-use for non-technical users
- speed improvement to tackle problems with multiple nonlinear functions
New perspective

Definition (Corridor)

Let \( h, l : [a, b] \to \mathbb{R} \) \( C^1 \) \( h(x) \geq l(x), \forall x \in [a, b] \) and having the same concavity. We call \( C = \{(x, y) | x \in [a, b], l(x) \leq y \leq h(x)\} \) a corridor between \( h \) and \( l \).
Convex corridor

Theorem (Convex corridor segment characterization)

On convex corridor $C$ there exists an optimal linear segment such that

- Both ends lie on the lower curve
- It is tangent to the upper curve
Convex corridor algorithm

generalisation of the algorithm of Ngueveu 2019

- Use dichotomy to find the tangent point
- find the resulting segment
- repeat 1 and 2 for the next segment

optimal 2-steps dichotomous search: No NLP or MINLP solved!

step a: \( \max \tilde{x} \)

step b: \( \max x_{\max_j} \)
Strength of the algorithm

- logarithmic convergence (for each segment)
- Works on concave corridors too
- not limited to absolute error
- can be used to approximate, underestimate and overestimate functions
Corridors Without Constant Convexity

Splitting the corridor

+ parallelizable
+ Efficient

- Heuristic Not necessarily optimal but the error is tightly bounded

\[ n^* \leq n \leq n^* + \#\text{Sub-corridors} \]

O’Rourke adaptation

based on function sampling and constraints on the line coefficient space

+ Exact

- Not as efficient
<table>
<thead>
<tr>
<th>function</th>
<th>absolute error</th>
<th>Continuous</th>
<th>ncc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>time</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>exact</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[RK15]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[RK19]</td>
<td></td>
</tr>
<tr>
<td>VI</td>
<td>0.1</td>
<td>Min 24.4 s</td>
<td>115 s</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>Few days 107.7 s</td>
<td>88 s</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>*</td>
<td>164 s</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>*</td>
<td>195 s</td>
</tr>
<tr>
<td>VII</td>
<td>0.1</td>
<td>* 311.1 s</td>
<td>226 s</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>* 14 514.6 s</td>
<td>287 s</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>*</td>
<td>268 s</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>*</td>
<td>869 s</td>
</tr>
<tr>
<td>VIII</td>
<td>0.1</td>
<td>sec 1.7 s</td>
<td>74 s</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>* 5.3 s</td>
<td>83 s</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>* 59.6 s</td>
<td>138 s</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>* 247.2 s</td>
<td>1466 s</td>
</tr>
<tr>
<td>IX</td>
<td>0.1</td>
<td>Few days 1.9 s</td>
<td>77 s</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>* 12 s</td>
<td>64 s</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>* 13 873.4 s</td>
<td>114 s</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>* 42 068.4 s</td>
<td>784 s</td>
</tr>
</tbody>
</table>
Results

Network design with congestion

- from a multivariate objective-function ... \( \min \sum_{(i,j) \in A} O_{ij} y_{ij} + \sum_{(i,j) \in A} \sum_{p \in P} D_{ij}^p x_{ij}^p + \sum_{i \in N} E_i z_i + \sum_{i \in N} g_i(\sum_{j \in N} \sum_{p \in P} x_{ij}^p, z_i) \)
- ... to a univariate objective-function

\[
\min \sum_{(i,j) \in A} O_{ij} y_{ij} + \sum_{(i,j) \in A} \sum_{p \in P} D_{ij}^p x_{ij}^p + \sum_{i \in N} f_i(v_i)
\]

<table>
<thead>
<tr>
<th>instance</th>
<th>literature</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>c35_0.3_0.6</td>
<td>6,615 s</td>
<td>6,92 s</td>
</tr>
<tr>
<td>c36_0.8_0.8</td>
<td>21,528 s</td>
<td>14,59 s</td>
</tr>
<tr>
<td>c49_0.8_0.6</td>
<td>172,255 s</td>
<td>118,17 s</td>
</tr>
<tr>
<td>c50_0.8_0.6</td>
<td>2609,568 s</td>
<td>2575,08 s</td>
</tr>
</tbody>
</table>

- as good as advanced state of the art solution methods
- no consideration on the problem structure
- easy to implement
Implementation

An approximation package named JULIA

Implementing both the exact and heuristic methods and include many classical error metrics!

- Julia based

\[ 2x^2 + x^3 + 10 \]

```
linearApprox((2x^2 + x^3 + 10), -2.5, 2.5, relative = true, epsilon = 1.0)
```

With an error guarantee!
<table>
<thead>
<tr>
<th></th>
<th>Motivation and key ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Continuous PWL approximation and absolute tolerance</td>
</tr>
<tr>
<td>3</td>
<td>nnc PWL approximation/bounding and relative tolerance</td>
</tr>
<tr>
<td>4</td>
<td>Corridors fitting and generalization to classes of errors</td>
</tr>
<tr>
<td>5</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Conclusion

Done

- PWLA+MILP efficient for solving certain classes of MINLPs
  - Non-necessarily continuous piecewise linear functions
  - Relative $\epsilon -$ tolerance
  - Bounding instead of approximation
- 2 similar MILPs to solve
- Various applications

What next?

- Coming soon (next few days) : opensource PWLB toolbox
- Extension to non-separable functions
- Other classes of problems ? (stochastic programming ?)
Bound each nonlinear function with 2 PWL functions

Modeling the PWL functions in a MILP

Solving MILPs containing PWL functions

DONE


Continuous piecewise linear delta-approximations for univariate functions:
Computing minimal breakpoint systems.

Piecewise linear function fitting via mixed-integer linear programming.

Global minimization of large-scale constrained concave quadratic problems by separable programming.

Piecewise-linear approximation with a bound on absolute error.
*Computers and Biomedical Research*, 7(1): 64 - 70.

Fitting piecewise linear continuous functions.
Useful Tools / Julia Packages

**LINA** : Computing a PWL approximation, over-/under-estimators with minimum # linear segments

- link: [http://homepages.laas.fr/sungueve/LINA.html](http://homepages.laas.fr/sungueve/LINA.html)
- input: a univariate continuous nonlinear function
- output: a nnc PWL function with minimum number of pieces
- related reference: Codsi, Gendreau, Ngueveu (2019-HAL)

**PiecewiseLinearOpt** : Modeling efficiently a given continuous PWL function in MILP

- [https://github.com/joehuchette/PiecewiseLinearOpt.jl](https://github.com/joehuchette/PiecewiseLinearOpt.jl)
- input: a continuous PWL function (or sampled nonlinear fct)
- output: variables and constraints to insert in a MILP