Mathematical Fidelity and Open Source Libraries for Large Scale Simulation and Optimization

William Symes

The Rice Inversion Project
Mathematics $\neq$ Computation, but...

Mathematical concepts $\rightarrow$ computational solutions

Example: function jets

$\Rightarrow$ Natural solutions - (1) storage and (2) update decisions for intermediate data, values
Agenda

Optimization

Jets

Optimization with Jets
Trust Region Algorithm

- $x = \text{vector, current estimate of minimizer}$
- $f = \text{objective function}$
- $g = \text{gradient of } f$
- $H = \text{approx. to Hessian of } f$
Trust Region Algorithm

- $s = \text{trial step}$
- $\Delta = \text{trust radius}$

Two major substeps in each iteration:

- constrained step computation
- step decision, constraint update
Trust Region Algorithm

Step Computation: given $\Delta, f, g, H,$

\[ s = \text{approx. minimizer of } \left( f + g^T s + \frac{1}{2} s^T H s \right) \]

subject to $s^T s \leq \Delta^2$

return $s$
Trust Region Algorithm

Step Decision: given $x, \Delta, s, f, f_+, g, H$, 

\[
\text{actred} = f - f_+ \\
\text{predred} = -g^T s - \frac{1}{2} s^T H s
\]
Trust Region Algorithm

Step Decision:

- if $\text{actred} < 0.1 \cdot \text{predred}$: $\Delta \leftarrow 0.5 \Delta$
- else $x \leftarrow x + s$
  - if $\text{actred} > 0.9 \cdot \text{predred}$: $\Delta \leftarrow 1.8 \Delta$

return $x, \Delta$
TR Algorithm: while (not done),

1. \( f = f(x), \ g = g(x), \ H = H(x) \)
2. \( s = \text{Step Computation}(f, g, H, \Delta) \)
3. \( f_+ = f(x + s) \)
4. \( x, \Delta = \text{Step Decision}(x, s, f, f_+, g, H, \Delta) \)
Agenda

Optimization

Jets

Optimization with Jets
Handling intermediate data

Heavyweight data common to
\( x \rightarrow f(x), g(x), \ldots:\)

- FEM computation: meshing, partitioning, elt. assembly, also used in \( g(x) \)
- time stepping: checkpointing, boundary data - adj. state method for \( g(x) \)
Handling intermediate data

Two problems: how to

1. share intermediate data, avoid recomputation
2. force recomputation when necessary

???
Handling intermediate data

obvious solution: pass required data structure as argument, i.e.

\[ f = f(x, stuff); \quad g = g(x, stuff) \]

(C programmer’s “void *” trick - TAO)

⇒ internal details of \( f, g, \ldots \) invade abstract algorithm
Handling intermediate data

from Diff’l Geometry: jet of function at point = (value, gradient, Hessian,...) - math. natural!

Computational jet: single data structure - value, gradient, intermed. data, ... = attributes

solves problem 1
Handling intermediate data

The Right Stuff (problem 2): end of TR step

\[ x, \Delta = \text{Step Decision}(x, s, f, f_+, g, H, \Delta) \]

at start of next step, has \( x \) changed (is “stuff” right)?
Handling intermediate data

Obvious solution: retain a copy $x_{\text{saved}}$ of $x$, check whether $x_{\text{saved}}$ same as $x$ on return

BUT

- generates more storage, data motion
- requires comparison of float data
Handling intermediate data

Efficient alternative: vector versioning

version is vector attribute, increment whenever vector assigned: for instance,

\[ x \leftarrow x + s \]

\[ \Rightarrow x.\text{version++} \Rightarrow \text{jets depending on } x \text{ must re-compute value, gradient,...} \]
Handling intermediate data

Jet semantics with vector versioning:

$$jet(f, x) = (f(x), g(x), ...)$$ for variable $x$

solves problem 2
Agenda

Optimization

Jets

Optimization with Jets
**TR with jets**

TR initialization, jet version:

- $x, x_+, s$: current iterate, trial iterate, step
- $e = \text{jet}(f, x), e_+ = \text{jet}(f, x_+)$
TR with jets

TR step, jet version: while (not done)

1. $s = \text{Step Computation}(e, \Delta)$
2. $x_+ = x + s$
3. $e, \Delta = \text{Step Decision}(e, e_+, \Delta)$
TR with jets

Step Decision, jet version: compute $actred$, $predred$ as before,

- if $actred < 0.1 \times predred$: $\Delta \leftarrow 0.5 \Delta$
- else $e \leftrightarrow e_+$
  - if $acted > 0.9 \times predred$: $\Delta \leftarrow 1.8 \Delta$

return $e, \Delta$
Summary

(Partial) Realizations - jets, similar stuff:

- Trilinos - NOX (jet=Group), ROL (jet=Objective)
- Rice Vector Library (jet=Evaluation)
- Madagascar algorithm dialect
- ...
Summary

Jets: mathematically correct, plus

- assure coherence of values, gradients,…
- side data (meshes, checkpoints,…) retained if possible, recomputed if necessary
- exactly the necessary value, gradient,… computations!
Thanks to:

- Jon Claerbout, Dave Nichols, Lester Dye, Amr El Bakry, Roscoe Bartlett, Sergey Fomel
- Mark Gockenbach, Shannon Scott, Hala Dajani, Tony Padula, Yin Huang
- The Rice Inversion Project, NSF