Two Functional MDD's for the Price of One - Part 2

Curtis d'Alves, Nhan Thai, Nassim Khoonkari,
Padma Pasupathi, Tanya Bouman, Christopher Anand

November 6, 2019

Outline

Symphony - Modeling Language for Non-Linear Optimization

- Models linear and non-linear optimization problems
- Simple declarative language
- Support for bounded parameters and constraint programming
- Generates performance oriented c code
- Solver Agnostic (plug into your solver of choice)

Vectors and dimension

- In Symphony, everything is vector.
- Vectors
  - Dimension (Shape): can be scalar, 1D, 2D, 3D, ...
    - ★ Scalar is just a single number
    - ★ 1D(n) variable is an array of n number, useful for problems in signal processing, sound processing, ..
    - ★ 2D(m x n) variable is a 2D array of m x n numbers, useful for problems in image processing, ...
    - ★ 3D(m x n x p) variable is a 3D array of m x n x p numbers, useful for problems in topology, image processing with voxels, ...
  - Numtype: can be real (R) or complex (C)
- We can manipulate vectors like adding, multiplying, doing inner product, ... to form new vectors (expressions).

Forming An Expression

- $(+), (-), (*), (/)$ Add/Subtract/Multiply/Divide (point-wise) two vectors having same shape and same numtype
- $(*.)$ Scale a vector with a scalar (if they form a vector space in Mathematics, i.e, real number can scale anything, but complex can only scale complex)
- $(<.>)$ Inner product (dot product) of two vectors
- $(\hat{})$ Power a vector with an integer
- Piecewise:

```
    case x:
      x <= 0     -> -x
      otherwise -> x

```

- sumElements, norm2square, normHuber

Structure

A valid symphony problem consists of:

- Variables
- Objective function
- Constants (optional)
- Constraints (optional)

Variable Declaration

- Variables are declared in a variable block
- For example:

```
variables:
  x[100][100] = 10
  y[20][20][20]
  a, b = 2, c

```

- Assignment denotes an initial value
- Unassigned variables will be randomly iniatlized witha number between (0,1)

Objective Function

- Declared in a minimize block
- For example:

```
minimize:
  (x - y)^2

```

- Must evaluate to a scalar (one dimensional value)

Constant Declarations

- Declared in a constants block
- For example:

```
constants:
  m = 2
   delta = 10, sigma = 15
   mask[100][100] = Pattern(FIRST_ROW_1)

```

- Unlike variables, cosntants must be assigned, and are not optimized over
- Multi-dimensional constants can be assigned using the Pattern function, which takes the following macros as input

```
    FIRST_COLUMN_0, FIRST_COLUMN_1, LAST_ROW_0
    LAST_ROW_1, LAST_COLUMN_0, LAST_COLUMN_1 ...

```

Local Variables

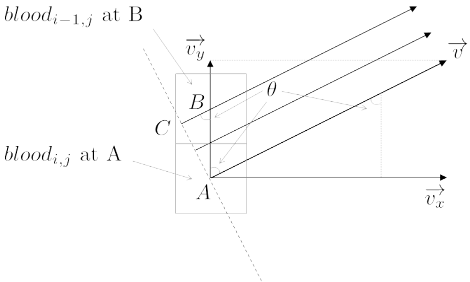
- Sometimes your expression can become to convoluted, declare local variables using a let block
- For example:

```
let:
  regularizerX = norm2square x
  regularizerY = norm2square y
  regularizer = regularizerX + regularizerY

minimize:
  norm2square (x - y) + regularizer

```

Sample Problem 1 - Velocity Problem



- MRI imaging problem dealing with blood flow
- Given vector field of blood flow: can we find how long each blood cell has been there?
- Do this by minimizing the flow over time (hence an optimization problem!)

Velocity Problem - Model Derivation

$$\Rightarrow t_{i-1,j} - t_{i,j} = \frac{CB}{|\overrightarrow{v}|} = \frac{AB\cos\theta}{\sqrt{v_x^2 + v_y^2}} = \frac{1\frac{v_y}{\sqrt{v_x^2+v_y^2}}}{\sqrt{v_x^2 + v_y^2}} = \frac{v_y}{v_x^2 + v_y^2} \tag{1}$$

$$\Rightarrow (t_{i-1,j} - t_{i,j})(v_x^2 + v_y^2) = v_y \tag{2}$$

$$\Leftrightarrow \Delta t_y(v_x^2 + v_y^2) = v_y \tag{3}$$

Velocity Problem - Optimization Model

$$\min_{t} \sum_{\text{pixels}} (\Delta t_x (v_x^2 + v_y^2) - v_x) * v_x^2)^2$$
$$+ \sum_{\text{pixels}} (\Delta t_y ((v_x^2 + x_y^2) - x_y) * x_y^2)^2$$

$v_{(x,y)}$	velocity in x,y direction
$t_{(x,y)}$	time in x,y direction

Variable Bounds

- Variable bounds are put in a constraints block
- For example:

```
constants:
  yUpperBound = 5
constraints:
  x >= 10
  y <= yUpperBound
```

- Bounds must be assigned to a value or constant (not an expression)

File Storage

- Values for variabels or constants can be loaded from text files or HDF5 datasets
- For example:

```
  constants:
    b[10][10] = File("b.txt")
  variables:
    real[128][128] = Dataset("dataset.hd5","real")
    imag[128][128] = Dataset("dataset.hd5","imag")

```

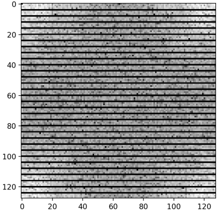
HDF5 - Hierarchical Data Format

- Designed to store and organize large amounts of data
- Capable of storing multiple labeled datasets in a single file
- Great library h5py for python, capable of generating data straight from numpy arrays

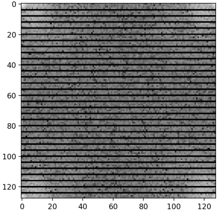
Brain Problem - 1

Data: real part (re) and imag part (im) of image's k-space received by the MRI.
Black spots are where the signal is lost.

re:

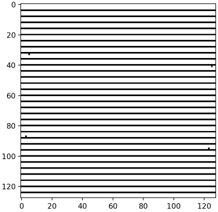


im:



Apply a threshold: signal = abs(re) > 0.5 to get a matrix of where the signal is received.
signal[i][j] = 1 if there is signal in this spot, 0 otherwise

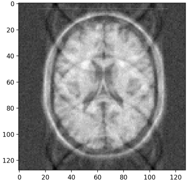
signal:



Brain Problem - 2

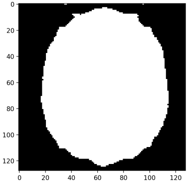
Naively reconstruct the image by taking inverse FFT, we get the naive image.

Naive image



median filter
threshold

median = normalize(ndimage.median_filter(img, 10))
head = median > 0.1



Determine the box constraint

x_lb and x_ub
- Inside head: pixel >= 0
- Outside head: -4 <= pixel <= 4

Multi-Coil MRI

$$\min_{\rho} \sum_{i=0}^{\#\text{coils}} ||FT(P)) - m_i||^2 + \lambda(||\delta_x(\rho)||^2 + ||\delta_y(\rho)||^2)$$

$$\begin{aligned}
\rho &= \text{True Image} \\
S_i &= \text{Coil Sensitivity} \\
m_i &= \text{K-Space measurement} \\
\lambda &= \text{scaling}
\end{aligned}$$

Constraints

- Declared in a constraints block (just like variable bounds)
- For example:

```
constraints:
  x <.> x >= 0
```

- Expression must be on LHS and must evaluate to scalar (one-dimensional value)
- **Note**: limits your choice of supported solvers

Logistics Problem

- In most logistic problems we want to Maximize the benefits or Minimize the costs.
- This table is for defining the revenue of sending products from factories to companies.

	Company1	Company2
Factory1	1.75	2.25
Factory2	2.00	2.50

- the below table shows the demand of companies. Also the capacity of each factory for producing is 60000.

	Company1	Company2
Factory1	x_{11}	x_{12}
Factory2	x_{21}	x_{22}
	23000	30000

Logistics Problem

- we want to Maximize the revenue.

$$Maxf(x) = Min - f(x)$$

- So we will have:

$$Min - ((c_{11} * x_{11}) + (c_{12} * x_{12}) + (c_{21} * x_{21}) + (c_{22} * x_{22}))$$
$$\text{subject to:}$$
$$x_{11} \geq 0,\ x_{12} \geq 0,\ x_{21} \geq 0,\ x_{22} \geq 0$$
$$x_{11} + x_{21} \geq \text{DemandCompany1}$$
$$x_{12} + x_{22} \geq \text{DemandCompany2}$$
$$x_{11} + x_{12} \leq \text{CapacityFactory1}$$
$$x_{21} + x_{22} \leq \text{CapacityFactory2}$$

Logistics Problem

variables here are the the products which are sent from each factory to each company.

$$\text{CapacityFactory1} = 600000$$
$$\text{CapacityFactory2} = 600000$$
$$\text{DemandCompany1} = 30000$$
$$\text{DemandCompany2} = 23000$$
$$c_{11} \rightarrow \text{revenue of } x_{11} = 1.75$$
$$c_{12} \rightarrow \text{revenue of } x_{12} = 2.25$$
$$c_{21} \rightarrow \text{revenue of } x_{21} = 2$$
$$c_{22} \rightarrow \text{revenue of } x_{22} = 2.50$$