Inference of rotational kinematics from accelerometer signals

Data-processing model

Contents

[General structure 1](#_Toc9777877)

[data structures 1](#_Toc9777878)

[input 1](#_Toc9777879)

[output 2](#_Toc9777880)

[window 2](#_Toc9777881)

[hyperparameters 2](#_Toc9777882)

[execution model 2](#_Toc9777883)

[function evaluation and differentiation 2](#_Toc9777884)

[relation to machine learning 2](#_Toc9777885)

[Case 1: “Horizontal plane, planar non-alignment” 4](#_Toc9777886)

[coordinate-transformation layer 4](#_Toc9777887)

[error function 5](#_Toc9777888)

[output: 6](#_Toc9777889)

[Other cases 8](#_Toc9777890)

[Case 1b: “Horizontal plane, non-alignment about tangential axis” 8](#_Toc9777891)

[Case 1c: “Horizontal plane, non-alignment about radial axis” 8](#_Toc9777892)

[Case 2: “Vertical plane, aligned” 8](#_Toc9777893)

[Case 2b: “Vertical plane, planar non-alignment” 9](#_Toc9777894)

[Appendix: answers to student exercises 9](#_Toc9777895)

[Case 1b: “Horizontal plane, non-alignment about tangential axis” 9](#_Toc9777896)

[Case 2b: “Vertical plane, planar non-alignment” 9](#_Toc9777897)

# General structure

## data structures

### input

*i* = 1 to N where *i* is the input position

where is the raw accelerometer sensor data

### output

*i* = 1 to M M <= N

the contents of vary from one model to another, but typically represent the time evolution of constraint-parameter values and of the rotational kinematic variables.

### window

The window selects a subset of the input data to process, generating a single output vector from it. It is the main processing element in the program. *K* consecutive input vectors treated in a single processing step. Other terms used analogously: “sliding window size”, “filter size”, “kernel size”.

### hyperparameters

* input length
* window size K, stride
* optimization parameters

## execution model

1. Initialize
   1. load data
   2. set hyperparameter values
   3. initialize window
2. window-based processing
   1. update constants of the function evaluator from the current window data.
   2. call the optimization routine.
   3. write output
3. terminate
   1. visualize input, output, performance stats
   2. write output to file

## function evaluation and differentiation

For these two capabilities we rely on the services provided by existing machine-learning libraries such as Theano or TensorFlow.

## relation to machine learning

To describe the data processing, some vocabulary and notation from current machine-learning architectures is borrowed ( CNN’s and RNN’s), but our learning model is very different in a number of important ways.

* the number of parameters in our models is very small ( anywhere from 2 to about 10 )
* both the parameters and the hidden variables correspond to human-interpretable, physically-based properties of a mechanical system.
* The system has no memory (outside of the current window). Based on a total error function for the current window position, we fully optimise the parameter values. The output, after each stride, is the (newly optimized) values the parameters.
* we do *not* rely on the usual operations such as convolutions or sigmoids. ( Modeling of the low-level signal conditioning, a later extension, would introduce convolution layer(s). )
* in our equivalent of a hidden layer, a hidden (vector) variable is computed *independently* for each input vector. ( There are no cross-connecting nodes in the computational graph ). The values parametrising the function used to do this, however, are the same for each input vector within the current window position. This provides a form of local regularization.

# Case 1: “Horizontal plane, planar non-alignment”



r

## coordinate-transformation layer

For each input element *k* = 1 to K within the current window position, we compute the *local kinematics*  associated with the point on the rigid body corresponding to the sensor location.

where is the 4-vector

composed of estimates for the radial acceleration , tangential acceleration , tangential velocity and anticipated change in velocity over the current time interval . Function *T* is composed of

and

where

The vector function is parametrised by the constraints parameters

whose values are determined though iterative minimization of an error function defined below.

( To avoid repeated calculations of the square-root operation over the iterations of the optimization loop, we will replace parameter *r* by and, after optimization is completed, use

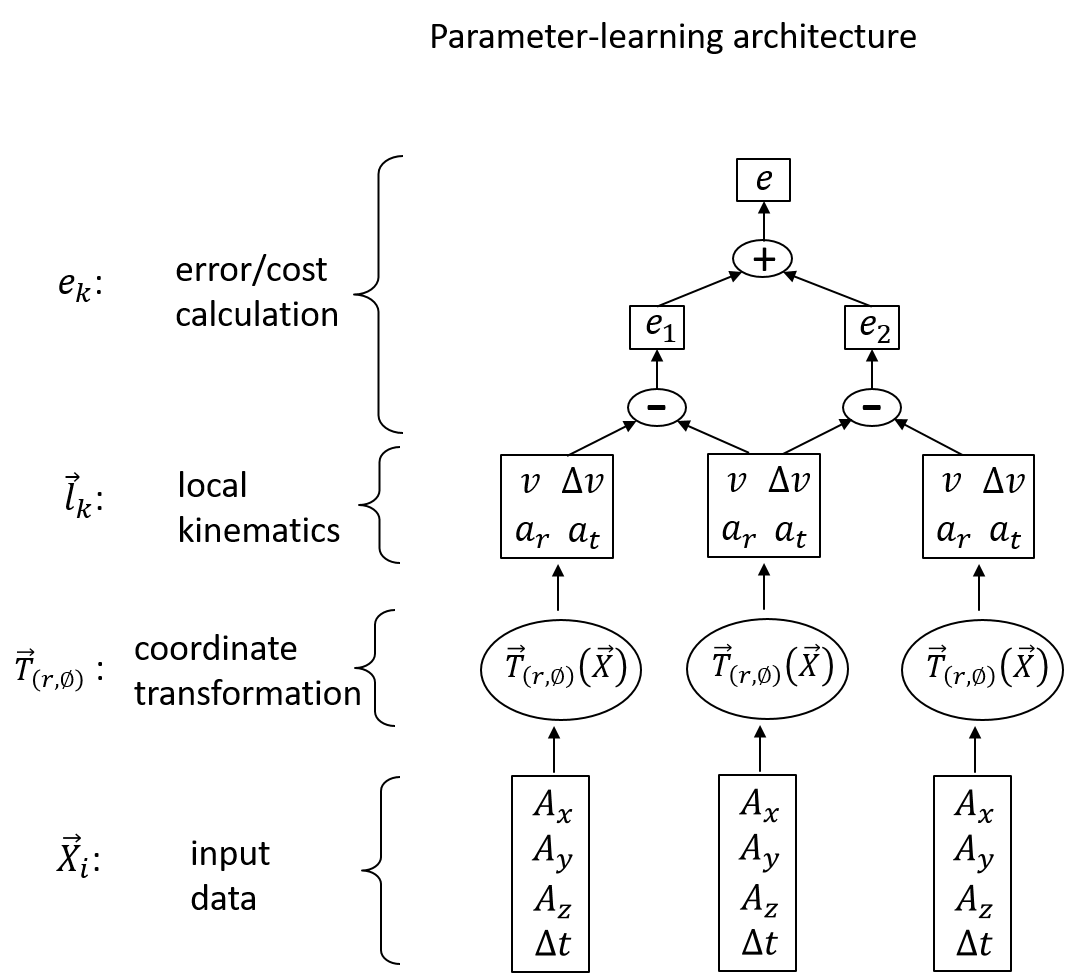
for the calculation of *v*. )

## error function

For each *j* = 1 to K-1 we predict the velocity associated with the next time step:

and the error resulting from a comparison with the velocity is computed from the input data of time step j+1:

The total error is the direct sum of the

The cost is minimized against the constraint parameters

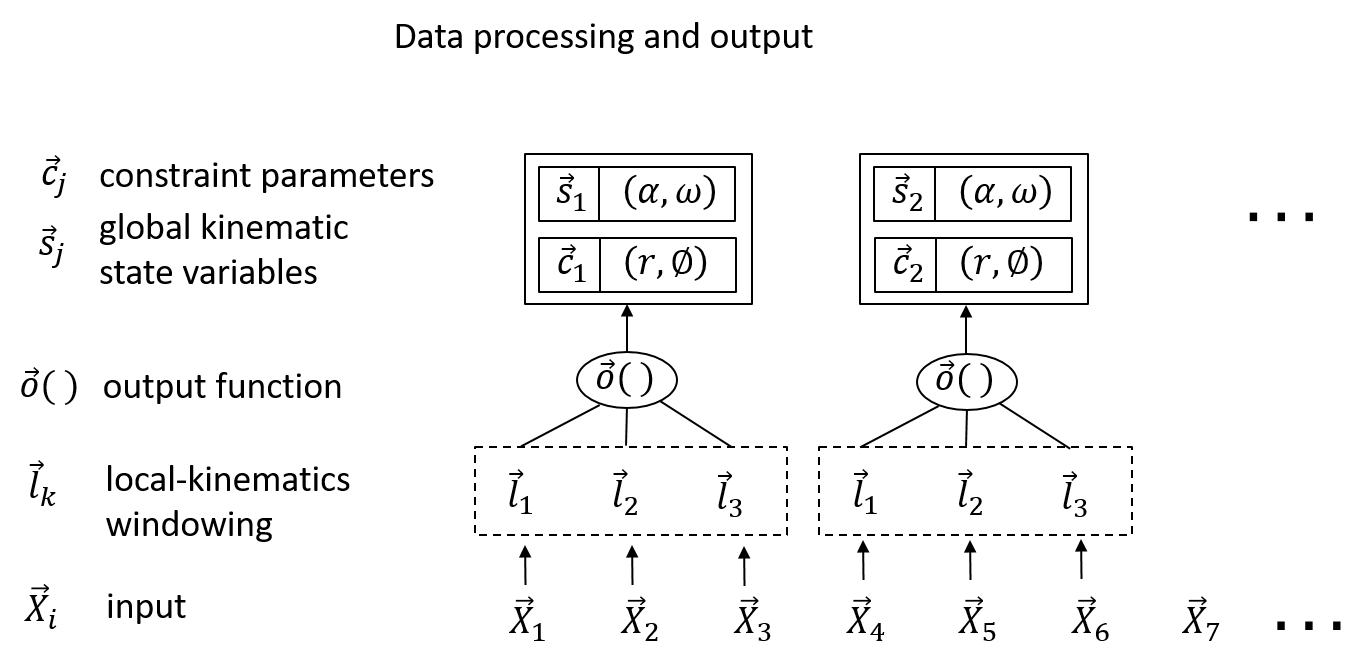
## output:

As the kernel strides along the input data, it produces an output at each window position

Outputs are new time-series sequences. There are two categories of outputs available:

1. constraint parameters discussed previously; and
2. motion or system-state variables where

where the overhead bar indicates an average over all K-1 values in the current processing window, and *r* is calculated from the optimization parameter via



# Other cases

## Case 1b: “Horizontal plane, non-alignment about tangential axis”

*r*

β

Fig. Side view of rigid body and sensor. Tangential axis points out of the page.

Student exercise.

## Case 1c: “Horizontal plane, non-alignment about radial axis”

Student exercise

## Case 2: “Vertical plane, aligned”

r

θ

g

Here we begin with a sensor whose x-axis is aligned with the radial (outwards) direction.

Figure 4 A system rotating in a vertical plane ( viewed from the side, i.e. viewing axis parallel to the Earth's surface )

Process is the same as before, only the function *T* is now with the following:

The calculations based on and proceed the same as before.

Optimization parameters are:

Output of state variable now includes as well:

## Case 2b: “Vertical plane, planar non-alignment”

Student exercise

# Appendix: answers to student exercises

## Case 1b: “Horizontal plane, non-alignment about tangential axis”

Here the sensor y-axis remains aligned with the tangential. We need only perform a rotation in the x-z plane to recover the radial component:

This should remove the gravitational component. One can verify this by evaluating the acceleration component perpendicular to the plane of rotation:

Its value should equal *g*.

## Case 2b: “Vertical plane, planar non-alignment”

Here, function T is

Optimization parameters are: . For a fixed axis, *r* and are constant, but varies . For non-fixed axes, the first two parameters can be treated as slowly varying or piecewise constant.