SPNG (de)convolution

Brett Viren

October 9, 2025

Topics

- New, general convolution kernels and operation in SPNG.
- Specific kernels for filtered-response deconvolution.
- Possible future application

Basic formalism

Fourier-space representation: convolution=multiplication, deconvolution=division.

Convolve signal S with response R to get measure M, in presence of noise N

$$M = S * R + N$$

Deconvolve measure M with response R subject to filter F to estimate signal S

$$S = F * M/R$$

Filter is required in order to (at least) suppress **divide-by-zero** but also to attenuate real-world **noise** N which is otherwise ignored in this decon model.

Deconvolution is convolution

Recognize an important associative property:

$$S = (F * M)/R = F * (M/R)$$

“Deconvolution” is really just convolution with a kernel that happens to have a denominator.

Basic convolution $C = T * K$ via FFT method

- If a dimension will have a **cyclic** convolution:

  ```
  shape[dim] = T.size(dim)
  ```

- If a dimension will have a **linear** convolution[1]:

  ```
  shape[dim] = T.size(dim) + K.size(dim) - 1
  ```

- Optionally, but only for linear, seek a **faster DFT size**:

  ```
  shape[dim] = faster_dft(shape[dim])
  ```

- **Pad** both T and K to have the calculated `shape`.
- The actual **convolution**:

  ```
  C = ifft2(fft2(T) * fft2(K)
  ```

- Optionally, but only for linear, **crop** away "faster" or additionally the "linear" padding.
- Optionally, **roll** dimension to account for any **"shifts"**.

[1] This avoids **cyclic artifacts** but the convolution is still cyclic.

Shifts (rolls) in C relative to T

Artificial shift

A badly prepared **kernel** can introduce a non-physical "shift".

- Eg, a mirror symmetry was not placed at sample zero.
- Best to "pre-shift" the kernel K before use, but can "post-shift" the convolution C.

Logical shift

The linear convolution is still cyclic and **early activity** can show up in **late samples**.

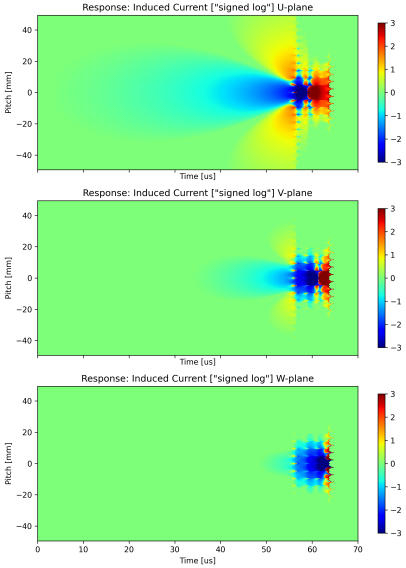
- Can "post-shift" C and then **relabel** sample zero with an earlier value (eg time).

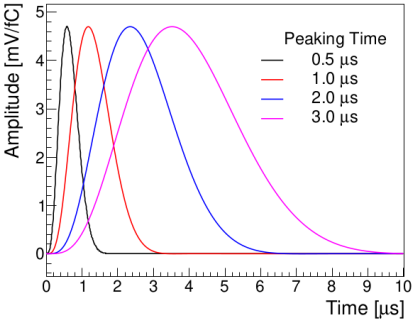
Natural shift

A kernel can naturally have a peak in a non-zero sample (FR and ER both have this).

- Peak in T will appear to have moved in C.
- This is **not a convolution error**, but due to the nature of the kernel contents.
- An application may apply a logical shift
  - eg, "we move the signal from the response plane to the collection plane"

Examples of shifts in FR and ER kernel parts





- FR's "pitch=0" row needs to be at tensor row=0 to avoid **artificial shift**.
- In $M = S * R$, both **naturally shift** peaks later in time.
- In $S = F * M/R$, natural shifts move peaks earlier in time. These can wrap around to place columns at high column number, causing a **logical shift**.

Real-world complications, F and R are composed from parts.

Responses

- 2D field response, calculated with a sample period that differs from ADC's.
- 1D electronics response, analytical function.
- 1D "RC" (or sometimes RC*RC) response with **long time constant** (large tensor).

Must combine these into one 2D R while also trying to exploit optimizations, especially w.r.t. 1D and long-time responses.

Filters

- 1D "wire" aka "channel" dimension filter.
- 1D pair of competing "time" filters ("gauss", "wiener").
- 1D multiple, competing additional filters for ROI finding (Lf/Hf, tight/loose).

Filters are an **arbitrary choice** and tend to made different compromises in **signal/noise** ratios and **charge conservation**.
Each subset of filters is needed for different purpose/consumer.

SPNG components - filter and response

`ITorchSpectrum` interface

Returns a Fourier-space tensor of a **given shape** and provides a **natural shape**.

`FilterKernel` provides F for decon

Combines N 1D filters with outer product to form ND filter.

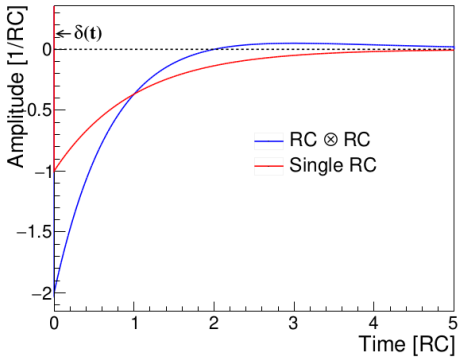
- Supports up to 3D, higher with simple code extensions.
- Intended for (channel, wire) filter.
- Directly reimplements WCT's analytical filter functions.
- Provides symmetric configuration for both low-pass and high-pass filters.

`ResponseKernel` provides R

Convolves FR and ER using configured standard components.

- Avoids **artificial shift** in FR channel dimension
- Downsamples FR to match ER period, linear convolution provides the "natural shape".
- Interval space padding (channel:central, time:end) + FFT to hit **requested shape**.
- Uses recently added `ThreadSafeCache` for LRU-cache of R as function of shape.

Q: What about RC? A: deal with it in a follow-on decon.



Typical: RC = 1 ms.

The RC response is very long, many milliseconds, $\mathcal{O}(10k)$ samples at 2 MHz. Best to handle it separately.

Performance

Including in the 2D R would require substantially more memory.

- The T tensor must also be padded to match
- Especially bloated for batched T.
- Avoid bloat with dedicated, serial, per batch or even per channel, 1D RC decon.

Physics

Future “chunked streaming” mode (supernova burst) requires special handling.

- Extra samples from convolution padding must be added to “next chunk”.
- Long RC may changes problem from “next chunk” to “next few chunks”.

SPNG components - `DeconKernel`

An `ITorchSpectrum` that provides the F/R kernel

F eg a `FilterKernel`

R eg a `ResponseKernel`

Very simple:

- Delegates to the two kernels to get each of a requested shape.
- Divides
- Optionally, caches the result.
  - Note, as can `ResponseKernel`. No use in caching in both.
- Returns the Fourier-space ratio.

This is NOT convolutional because the filter is assumed to itself be a sampling in Fourier-space.

- No extra padding is introduced beyond what is in the R denominator.

SPNG components - `KernelConvolve`

An `ITorchTensorFilter` that applies $C = T * K$ convolution
Per-dimension options

- Apply cyclic (no padding) or linear convolution.
- Crop C by fixed amount, to remove “faster” padding or additionally “linear” padding.
  - only if linear
- Apply a roll to C after possible cropping.

Status and next steps

Current status

All described here exists and with fairly good unit test coverage.

- With Gemini, etc, there's really no excuse not to have unit tests!

Near future

Finish Jsonnet config to do integration testing of `KernelConvolve` and friends.

- Check that the actual results are reasonable.

Next next things

1. Add the RC decon follow-on node. Maybe can extend `KernelConvolve` to handle this 1D convo.
2. Retrofit and/or integrate the existing MP2/3 node.

Far in the future, but becoming a realistic goal.

It recently occurred to me that we are actually providing some of the "hard parts" of TPC detector simulation. SPNG now has perhaps 20% of what is needed.

Other changes along the way

`util/`

- Thread safe caches
- JSON $\leftrightarrow$ C++ struct via Boost.Hana

`spng/`

- Ported WCT's LMN to Torch
- Implement `TorchSetUnpacker` giving tensor set -> tensor fanout.
- Use thread safe cache for kernel tensors
- Use Boost.Hana support for better configuration.

New, robust C++-layer configuration pattern

Standard WCT C++ configuration sucks. Interpreting JSON object is error prone, not standardized and is verbose. Boost.Hana gives key to a better way. No new dependencies required.

```
struct MyComponentConfig {
    int number = 42;
    std::string tool = "";
};
BOOST_HANA_ADAPT_STRUCT(MyComponentConfig, number, tool);  // some magic
class MyComponent : public ISomeInterface {
    MyComponent m_cfg;          // use C++ struct, not JSON!
    // ...
};
void MyComponent::configure(const WireCell::Configuration& config) {
    from_json(m_cfg, config);   // type safe conversion
}
WireCell::Configuration MyComponent::default_configuration() const {
    return to_json(m_cfg);      // and all automatic, thanks hana.
}
void MyComponent::use_my_config() {
    std::cout << "number=" << m_cfg.number << " tool=" << m_cfg.tool << "\n"
}
```

More improvements possible but requires radical schema-based approach a'la moo.