Xobjects and Xdeps: Low-Level Libraries Empowering Beam Dynamics Simulations

S. Łopaciuk, R. De Maria, G. Iadarola

Beams Department, CERN, Geneva, Switzerland

Xsuite – A Beam Dynamics Simulation Package [1]

- As a **Python** package, it is easy to use and flexible.
- Fast and extensible, since critical parts are in **C**.
- **Multi-threading** on CPU (OpenMP [2]), **GPU** (CUDA [3], OpenCL [4]).
- Imports from MAD-X, includes **expressions**.

Powered by **Xobjects** and **Xdeps**.

Xobjects – An In-Memory Serialiser

Writing portable GPU-accelerated code is non-trivial.

- With Xobjects, **write code once**.
- Support single- and multi-threading.
- OpenMP, CUDA, OpenCL.

Specify **portable binary objects** for the code.

- Accessible from Python and C.
- Standard numeric types supported.
- Multi-dimensional arrays with any strides, fixed/dynamic shapes.
- Compound types: structs, unions, etc.

An Xobject Example

- Represent an array of $\mathbb{R}^2$ vectors:

```
import xobjects as xo

class Vectors(xo.Struct):
  x = xo.Float64[:]
  y = xo.Float64[:]
```

- Instantiate and access an Xobject from Python:

```
vec = Vectors(x=[1, 2], y=[3, 4])
vec.x.to_nparray()  #=> np.array([1, 2])
```

- Xobjects will generate a C API, for defining accelerated functions:

```
int64_t Vectors_len_x(Vectors obj);
double Vectors_get_x(const Vectors obj, int64_t i0);
double Vectors_set_x(
  const Vectors obj,
  int64_t i0,
  double value);
// etc...
```

- With the API own functions can be defined, e.g. we can calculate lengths of element vectors in parallel (`ArrayNFloat64 = xo.Float64[:]`):

```
/*gpufun*/ void get_lengths(
    VectorsData v,
    ArrayNFloat64 result
) {
    int64_t n = Vectors_len_x(p);
    int64_t i = 0;
    //vectorize_over i n
    double x, y, l;
    x = Vectors_get_x(v, i);
    y = Vectors_get_y(v, i);
    l = sqrt(x*x + y*y);
    ArrayNFloat64_set(result, i, l);
    //end_vectorize
}
```

- The decorations tell Xobjects how to parallelise the code.

Use in Particle Tracking

- Non-collective line elements have **parallelisable tracking functions**.
- Track sections of the beam line in parallel, or in sequence, as needed.
- Portability: data can easily **move between CPU and GPU** contexts.
- Creating own elements is as simple as subclassing, **no knowledge of internals needed**!

Xdeps – A Data Dependency Manager and Optimiser

Implement MAD-X deferred expressions ($a := b$) in Xsuite.

- Declare **variables** and their **dependencies**.
- Keep values **updated** based on the dependencies.

An **optimiser** module used for **optics matching**.

Design

Xdeps can be used to update **any Python object**. Key concepts:

- **Manager** – **register** external objects, orchestrate **actions**.
- **Task** – describes an **action** that modifies the values of **targets** according to a set of **dependencies**, and potentially its internal state.
- **Expression** – represents a **generic Python expression** between different values. Can define a task when an expression of references is assigned to another reference.

An Xdeps Example

- Set-up: convert Cartesian to planar coordinates.

```
import xdeps as xd, math
from types import SimpleNamespace

pt = SimpleNamespace(x=1, y=1)

# Set up manager and reference
mgr = xd.Manager()
pt_ = mgr.ref(pt, 'pt')
math_ = mgr.ref(math, 'math')

# Define expression-based tasks
pt_.r = math_.sqrt(pt_.x**2 + pt_.y**2)
pt_.th = math_.atan2(pt_.y, pt_.x)
```

- Updating x recomputes r and θ:

```
pt_.x = 0
print(pt.x, pt.r, pt.th)  #=> (0, 1.0, pi/2)
```

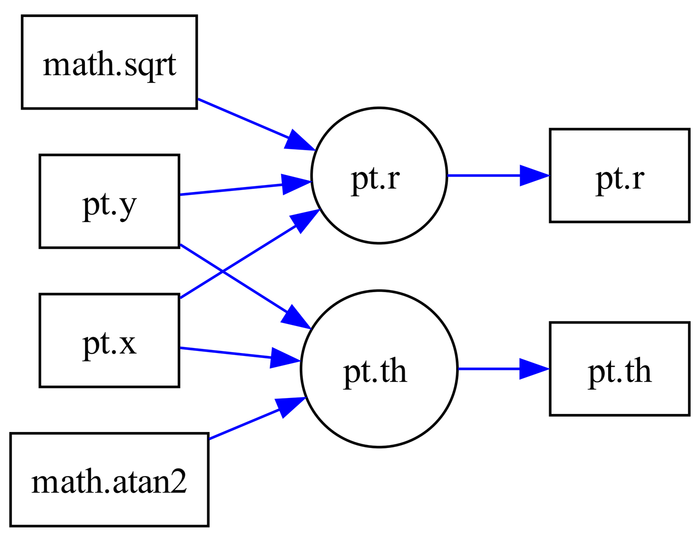

Figure 1: The graph of expressions (boxes) which are dependencies and targets of tasks (circles). Can be plotted with `mgr.plot_deps()`.

An Optimiser Suited for Fine-Tuning

- Tried-and-tested **Jacobian** optimiser of MAD-X [5].
- To start, specify target **conditions**, **tolerances**, **weights**, and **parameters** to vary; **defaults** can be given.
- Match typically against Twiss, but any observable can be chosen.
- A summary of the match results can be printed, and a **log of the iteration steps** is available.
- The match state can be **rolled back**, and targets or parameters can be **enabled or disabled**.
- **Single steps** of the match can be executed, if necessary.

References

[1] G. Iadarola, et al., Xsuite: An Integrated Beam Physics Simulation Framework. *Proceedings of the 68th ICFA ABDW on High-Intensity and High-Brightness Hadron Beams*, Geneva, Switzerland, 2023.

[2] L. Dagum and R. Menon, OpenMP: An Industry Standard API for Shared-Memory Programming. *IEEE Computational Science and Engineering*, 5(1):pp. 46–55, 1998.

[3] J. Nickolls, et al., Scalable Parallel Programming with CUDA. *ACM SIGGRAPH 2008 Classes*, pp. 1–14, ACM, Los Angeles California, 2008.

[4] J. E. Stone, D. Gohara, and G. Shi, OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *Computing in Science & Engineering*, 12(3):pp. 66–73, 2010.

[5] R. de Maria, F. Schmidt, and P. K. Skowronski, Advances in Matching with MAD-X. *Proceedings of ICAP*, pp. 213–215, 2006.