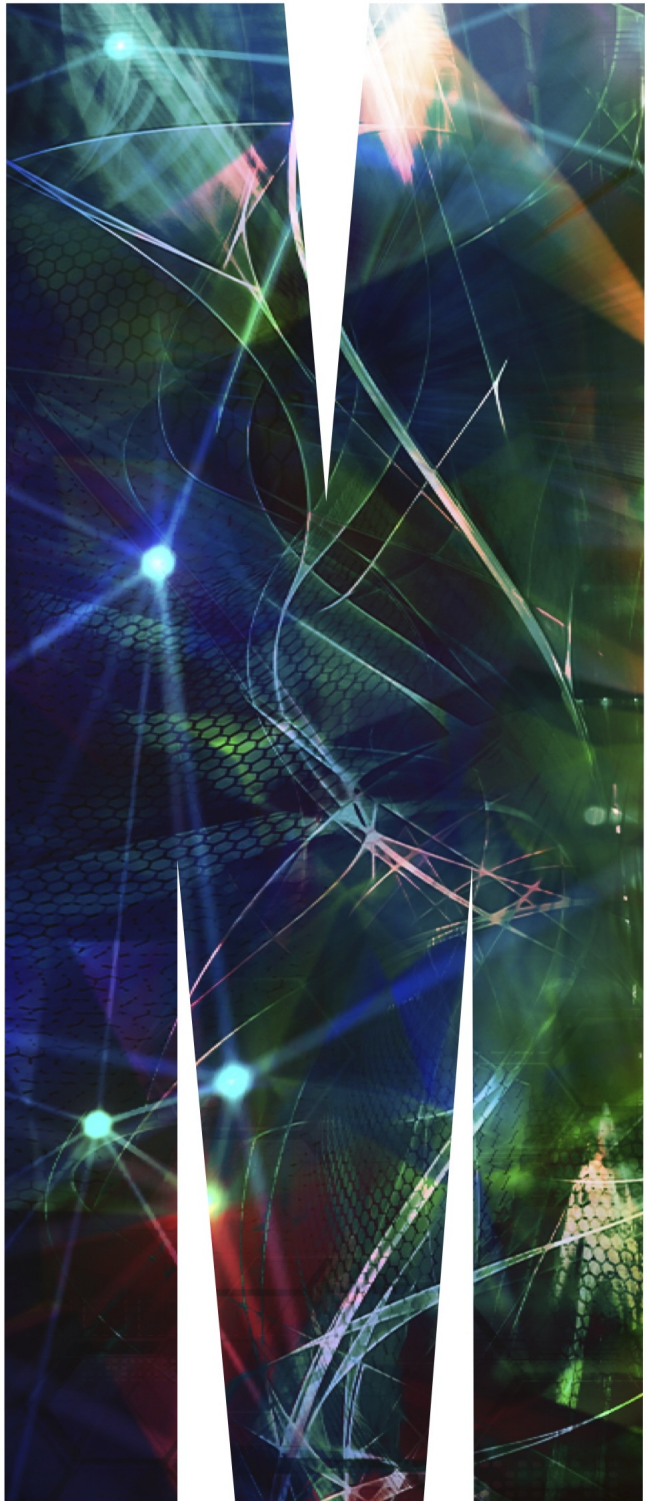
External Matrix-Element Interfaces

Christian T Preuss

April 13, 2021



Current status and ideas for the future

- both VINCIA and DIRE have used (different) MadGraph5 plug-ins for matrix-element corrections
- has (partially) been standardised during integration of VINCIA and DIRE into 8.3
- as of 8.304 both showers can use (pre-generated) MadGraph5 matrix-element libraries via a single run-time interface
  - release contains a basic set of processes, (compilation) enabled by `./configure -with-mg5mes`
  - additional libraries can be generated using the `generate` script in `plugins/mg5mes/` using standard MG5 syntax, e.g.:

```
./generate --process="p p > e+ e-" --output=pp2ee
```

  - can be loaded at run time, e.g. `Vincia:MEplugin = pp2ee`

Ideas:

- consolidate and simplify existing structure as much as possible
- make it simple to use framework in other parts of PYTHIA (tau decays, BSM, ... ?)
- streamline developments
- prepare for extensions to further ME generators

Current status – implementation

- **generic** base class `ShowerMEs`
- actual MG5 interface in derived class `ShowerMEsMadgraph`

```
class ShowerMEsMadgraph : public ShowerMEs {

public:

  // Constructor.
  ShowerMEsMadgraph() {isInitPtr = false; isInit = false;
    libPtr = nullptr; modelPtr = nullptr;}

  // Destructor.
  ~ShowerMEsMadgraph() {if (libPtr != nullptr) delete libPtr;
    if (modelPtr != nullptr) delete modelPtr;}

...

private:

  PY8MEs_namespace::PY8MEs* libPtr;
  PARS* modelPtr;

};
```

- loaded at **run time** via `ShowerMEsPlugin` class

```
class ShowerMEsPlugin : public ShowerMEs {

...

private:

  // Typedefs of the hooks used to access the plugin.
  typedef ShowerMEs* NewShowerMEs();
  typedef void DeleteShowerMEs(ShowerMEs*);

  // The loaded MEs object, plugin library, and plugin name.
  ShowerMEs *mesPtr;
  PluginPtr  libPtr;
  string     name;

};
```

Consolidate and extend

- despite standardisation, still some code duplication we could avoid, cf. `ShowerMEs`:

```
// VINCIA methods.
// Set pointers to required PYTHIA 8 objects.
virtual void initPtrVincia(Info* infoPtrIn, SusyLesHouches* slhaPtrIn,
  VinciaCommon* vinComPtrIn);
// Initialise the MG5 model, parameters, and couplings.
virtual bool initVincia() = 0;
// Check if the process is available.
virtual bool hasProcessVincia
(vector<int> idIn, vector<int> idOut, set<int> sChan) = 0;
// Get the matrix element squared for a particle state.
virtual double me2Vincia(vector<Particle> state, int nIn) = 0;
// Use me2 to set helicities for a state. Takes a reference as
// input and operates on it.
virtual bool selectHelicitiesVincia(vector<Particle>& state, int nIn,
  bool force);
// Set and get colour depth.
virtual void setColourDepthVincia(int colourDepthIn) {
  colourDepth = colourDepthIn;}
virtual int getColourDepthVincia() {return colourDepth;}
// Convert a process label to a string, e.g. for printing to stdout.
string makeLabelVincia
(vector<int>& id, int nIn, bool convertToNames = false) const;
// Set verbosity level.
virtual void setVerboseVincia(int verboseIn) {verbose = verboseIn;}

// Dire methods.
virtual bool initDire(Info* infoPtrIn, string card) = 0;
virtual bool isAvailableMEDire(vector <int> in, vector<int> out) = 0;
virtual bool isAvailableMEDire(const Pythia8::Event& event) = 0;
virtual double calcMEDire(const Pythia8::Event& event) = 0;
```

⇒ identify use cases, common structures, … (and drop `Vincia`/`Dire` suffixes)

Consolidate and extend

- `ShowerMEs` base class is already very generic, but could be prepared for general ME generator interfaces (and more general use cases in PYTHIA)
  - e.g. not every generator supports helicity or colour-ordered amplitudes
  - rename `ShowerMEs` → `ExternalMEs` (or similar)?
  - extension to loop providers?
    (private interface to MCFM under development, but also OPENLOOPS or MG5 possible)
- enable loading of multiple MG5 libraries (in `ShowerMEsMadgraph` or `ShowerMEsPlugin`)

  ⇒ allows to have multiple `ShowerMEsPlugin`s, each containing multiple processes, e.g.:

  ```
  Vincia:MEplugins = {mg5, sherpa}
  MG5Interface:Libraries = {pp2ee, pp2ev}
  ```
- new interface to SHERPA?

SHERPA/COMIX interface

- SHERPA has two built-in ME generators: AMEGIC and COMIX
- COMIX uses colour-dressed Berends-Giele recursion relations
  - ⇒ extremely fast amplitude evaluation
  - ⇒ fully automated (no pre-run generation/compilation of libraries needed)
- interface to AMEGIC in principle possible, but no need (so far?)
- interface builds on **well-tested** components:
  - **Pythia side**: `ShowerMEs` + `ShowerMEsPlugin` structure
  - **Sherpa side**: COMIX interface used in SHERPA resummation plug-in (1411.7325, 1912.09396)

Caveats:

- interface somewhat tied to SHERPA-internal developments
  (need to synchronise release / provide Docker image ?)
- only SM implemented and tested so far

⇒ Is there interest in having this interface/making it public?