The Belle II analysis software framework **basf2** has a modular design with Python steering of on-demand dynamically-loaded C++ modules and event-based parallel-processing capability.

---

**The Belle II experiment** at the SuperKEKB colliding-beam $e^+e^-$ accelerator in Tsukuba, Japan, studies the behaviour and symmetry properties of heavy quarks and leptons.

---

**GEANT4** – records crossings at each GEANT4 sensitive-volume boundary
- assumes 6 particle-type hypotheses ($e$, $\mu$, $\pi$, $K$, $p$, $d$)
- extrapolates each reconstructed charged track outward

Two basf2 event-reconstruction modules for particle ID using **GEANT4E**
- **ext**
  - extrapolates each reconstructed charged track outward
  - assumes 6 particle-type hypotheses ($e$, $\mu$, $\pi$, $K$, $p$, $d$) per track
  - records crossings at each GEANT4 sensitive-volume boundary
  - records time, position, momentum, phase-space covariance matrix

- **muid**
  - extrapolates each track through the $K_\mu$–muon detector (KLM)
  - assumes only the muon hypothesis
  - applies a Kalman filter at each layer crossing; adjusts the extrapolated-track properties based on matching-hit location
  - assigns particle-identification likelihood based on KLM’s measured-vs-extrapolated range + transverse scattering

---

**GEANT4 and GEANT4E coexistence in basf2**

**GEANT4E**, as distributed, cannot be used with GEANT4:
- incompatible particle lists
- incompatible physics processes
- conflicting usage of sensitive-detector geometry
- distinct states when calling RunManager
- distinct step-by-step Navigators
- incompatible user actions (SteppingAction etc)

**GEANT4E**, as distributed, is limited:
- propagates only electrons, positrons and photons

We have resolved these issues and limitations. All of our modifications are done **outside** the **geant4** code base.

---

- The distributed G4EnergyLossForExtrapolator defines energy-loss processes for electrons and positrons only.
- Our custom G4EnergyLossForExtrapolator extends these processes to $\mu$, $K$, $p$ and $d$ (and their anti-particles).
- During GEANT4 simulation, G4SteppingManager calls user code to process steps through “sensitive” detector volumes and record the hits therein.
- During GEANT4 extrapolation, our custom version of StepLengthLimitProcess() disables this behaviour:

```
G4SteppingManager* mySteppingManager = ...;
mySteppingManager->SetUserAction(new MySteppingAction());
```

- Avoid the special G4ErrorPropagationNavigator in GEANT4E. Instead, use the standard G4Navigator defined in GEANT4.
- GEANT4E requires a **target surface** (G4ErrorCylSurfaceTarget is an infinite-length cylinder). After each GEANT4E step, G4ErrorPropagationNavigator would have checked if track crossed this surface. Our steering code does this check.
- Our custom version of G4ErrorCylSurfaceTarget is a closed finite-length cylinder that includes the endcap surfaces.
- The distributed MagFieldLimitProcess in GEANT4E assumes that the magnetic field is along the $z$ axis. Our custom MagFieldLimitProcess removes this assumption.