

Comments from JSF users

Katsumasa Ikematsu (KEK)

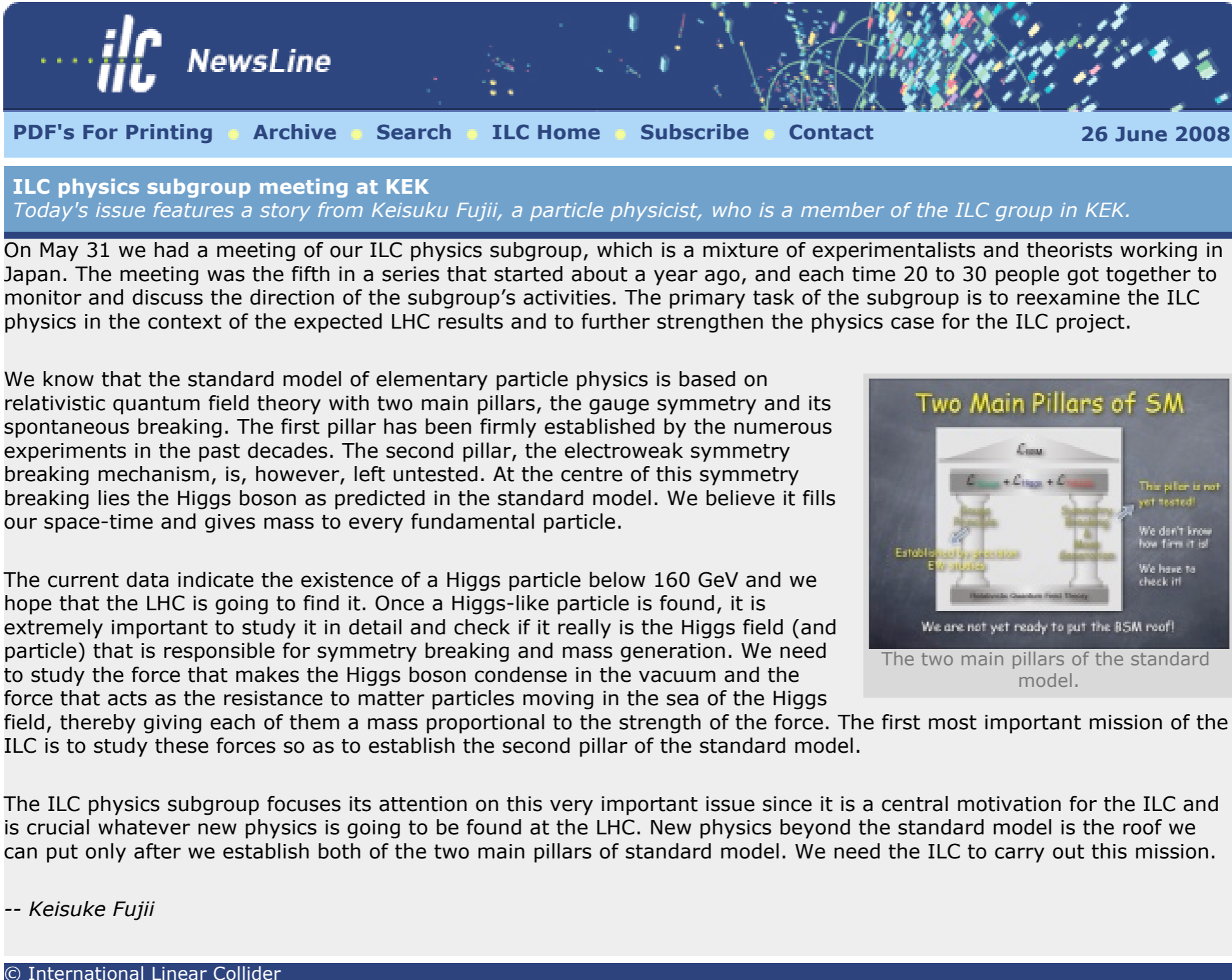
ILD Software Workshop (16/Apr/'09 @KEK)

Situation in Asia (mainly in Japan)

- ILC physics working subgroup activities
 - Mixture of 20~30 experimentalists & theorists from Japan & China
 - EWSB physics: $ZHH@500\text{GeV}$, $TTH@500\text{GeV}$, $\gamma\gamma \rightarrow HH$ / ($H \rightarrow bb, WW^*$)
 - New physics / cosmological connection: LHT w/ T-parity, Generic WIMP studies, anomalous SM couplings etc. => lots of talks in the past & coming LC workshops
 - Basically, independent of ILD-LOI physics performance studies
 - **Fast detector simulator** (JSFQuickSim)
 - **ROOT-based analysis libraries** (LEDA - “Library Extension for Data Analysis”)
- JSF as a main study framework for the subgroup activities
 - Needs to keep their momentum using “ILD software”
 - Possible solution: **Fast simulator implementation in the Marlin framework?**

Situation in Asia (mainly in Japan)

- ILC physics working subgroup activities



ilc NewsLine

PDF's For Printing • Archive • Search • ILC Home • Subscribe • Contact 26 June 2008

ILC physics subgroup meeting at KEK

Today's issue features a story from Keisuke Fujii, a particle physicist, who is a member of the ILC group in KEK.

On May 31 we had a meeting of our ILC physics subgroup, which is a mixture of experimentalists and theorists working in Japan. The meeting was the fifth in a series that started about a year ago, and each time 20 to 30 people got together to monitor and discuss the direction of the subgroup's activities. The primary task of the subgroup is to reexamine the ILC physics in the context of the expected LHC results and to further strengthen the physics case for the ILC project.

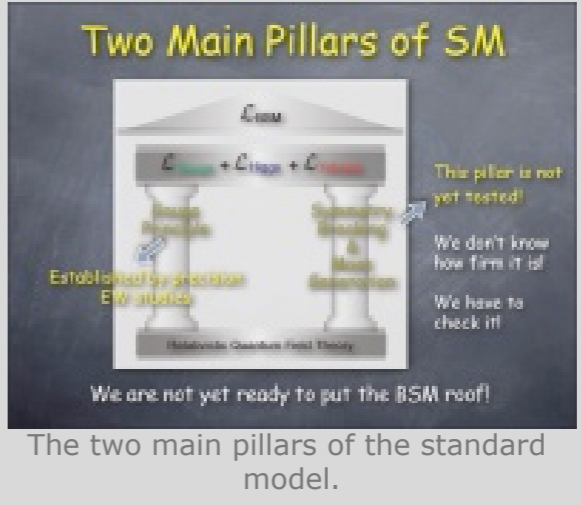
We know that the standard model of elementary particle physics is based on relativistic quantum field theory with two main pillars, the gauge symmetry and its spontaneous breaking. The first pillar has been firmly established by the numerous experiments in the past decades. The second pillar, the electroweak symmetry breaking mechanism, is, however, left untested. At the centre of this symmetry breaking lies the Higgs boson as predicted in the standard model. We believe it fills our space-time and gives mass to every fundamental particle.

The current data indicate the existence of a Higgs particle below 160 GeV and we hope that the LHC is going to find it. Once a Higgs-like particle is found, it is extremely important to study it in detail and check if it really is the Higgs field (and particle) that is responsible for symmetry breaking and mass generation. We need to study the force that makes the Higgs boson condense in the vacuum and the force that acts as the resistance to matter particles moving in the sea of the Higgs field, thereby giving each of them a mass proportional to the strength of the force. The first most important mission of the ILC is to study these forces so as to establish the second pillar of the standard model.

The ILC physics subgroup focuses its attention on this very important issue since it is a central motivation for the ILC and is crucial whatever new physics is going to be found at the LHC. New physics beyond the standard model is the roof we can put only after we establish both of the two main pillars of standard model. We need the ILC to carry out this mission.

-- Keisuke Fujii

Two Main Pillars of SM



The two main pillars of the standard model.

Full simulator & Fast simulator



- Use both drivability (its pros and cons) as the situation demands
 - ZHH team & TTH team intend to move to the FullSim framework (taking advantage of the **powerful mass production** - 500GeV SM BGs + providing additional signal StdHep files)
 - New physics team & $\gamma\gamma$ team continue to use FastSim framework (taking advantage of the **light weight & non resource-consuming** programs)
 - ▶ e.g. $\gamma\gamma \rightarrow WW \rightarrow 4\text{-jets}$ ($\sim 90\text{pb}$) BG against $\gamma\gamma \rightarrow HH$ ($\sim 0.5\text{fb}$)
- **If our resources allow we want to have both!**
 - **Same interface** preferable
 - Should be **easy to handle** for both

Migration to the Marlin world?

- JSF is a **framework for event-by-event data analysis**
 - Provides a framework for modular analyses suitable for “event generation - det. simulation / testbeam data - analysis” chain
 - Same as the Marlin world => expects no big interference (technically)
- But, JSF is **based on ROOT**
 - Usefulness of the ROOT-CINT
 - ▶ Unified framework for **interactive and batch jobs** => efficient .C macro development environment
 - Configuration file similar to .rootrc is used to set parameters
 - User defined command line argument => input values can be overridden at run time => useful for batch run
 - **Object I/O**
 - ▶ Data as branches of a ROOT tree can be saved/read in each modules
- Possible to **realize similar culture** in the Marlin world??

A fast det. simulator: JSFQuickSim

- Major member functions

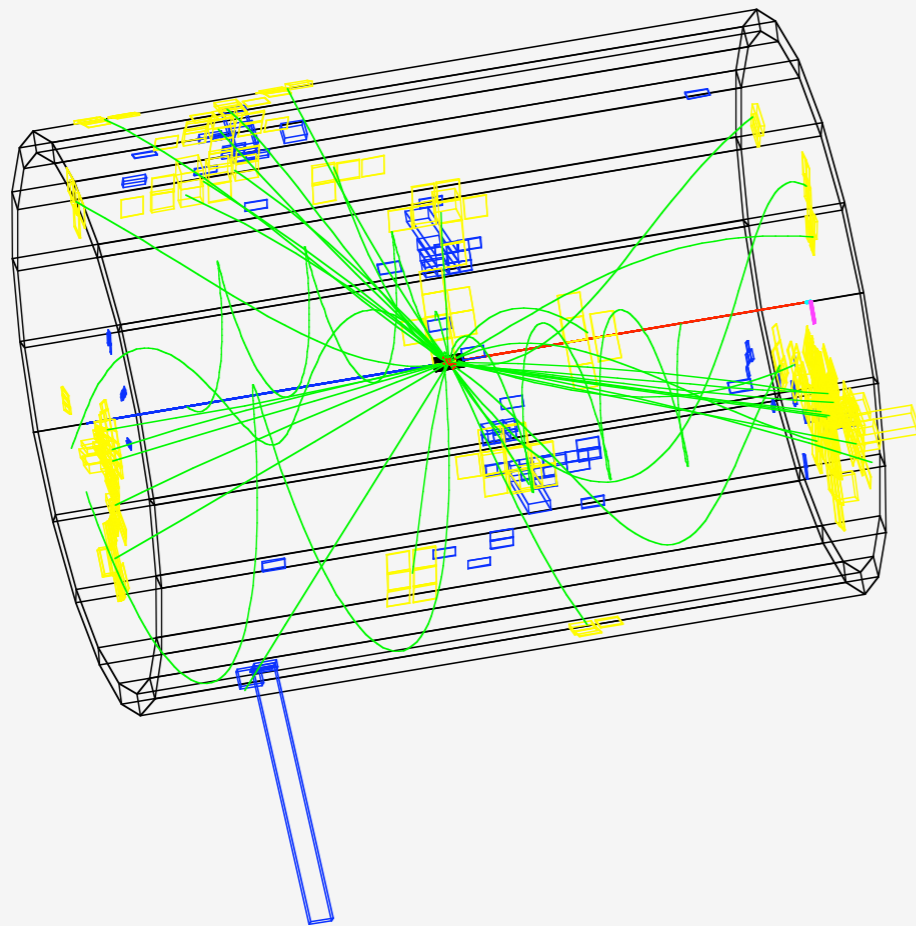
```
Int_t GetNLTKCLTracks();  
Int_t GetNCDCTracks();  
Int_t GetNVTXHits();  
Int_t GetNEMCHits();  
Int_t GetNHDCHits();  
Int_t GetNSMHits();  
Int_t GetNGeneratorParticles();
```

```
TObjArray *GetLTKCLTracks(); // Pointers to LTKCLTracks objects array  
TClonesArray *GetCDCTracks(); // Pointers to CDCTracks object array  
TClonesArray *GetVTXHits(); // Pointers to VTXhits object array  
TClonesArray *GetEMCHits(); // Pointers to EMhits object array  
TClonesArray *GetHDCHits(); // Pointers to HDhits object array  
TClonesArray *GetSMHits(); // Pointers to SMhits object array  
TClonesArray *GetGeneratorParticles(); // Pointers to GeneratorParticle  
objects array
```

$e^+e^- \rightarrow ttH$ event display

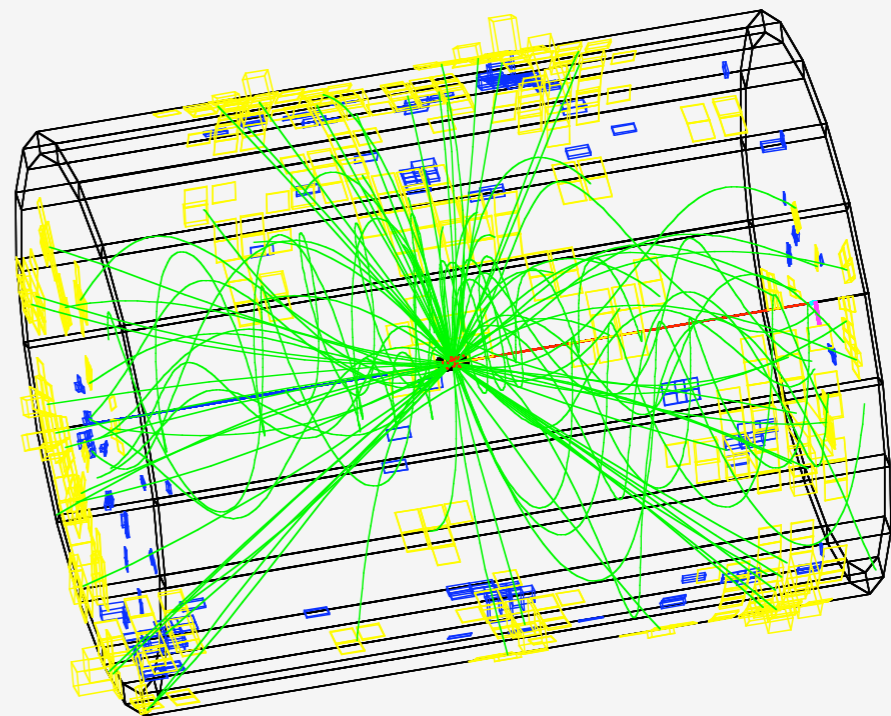
- Dense 8-fermion ($H \rightarrow bb$) & 10-fermion ($H \rightarrow WW^*$) events
- Challenging for correct reconstruction (jet-association)

Lepton + 6-jets mode



$ttH \rightarrow (bW)(bW)(bb) \rightarrow (bud)(bl\nu)(bb)$

8-jets mode



$ttH \rightarrow (bW)(bW)(bb) \rightarrow (bcs)(bcs)(bb)$

Combined track: JSFLTKCLTrack

- End-user mainly uses track/CAL-cluster linked (= energy flow) tracks only
- Major data members

```
Double_t    fP[4];    // four momentum (E,Px,Py,Pz), GeV
JSFCDCTrack *fCDC;    //! Address of corresponding CDC track.
Float_t     fVTXNSig; // Sqrt( (VTXDR/VTXDDR)^2 + (VTXDZ/VTXDDZ)^2 )
Int_t       fType;    // ITYP
```

- 1: Pure gamma,
- 2: Gamma in mixed EMC,
- 3: Pure neutral Hadron,
- 4: Hadron in mixed HDC,
- 5: Pure charged hadron,
- 6: Unmatched track,
- 11: Electron candidate,
- 13: Muon candidate

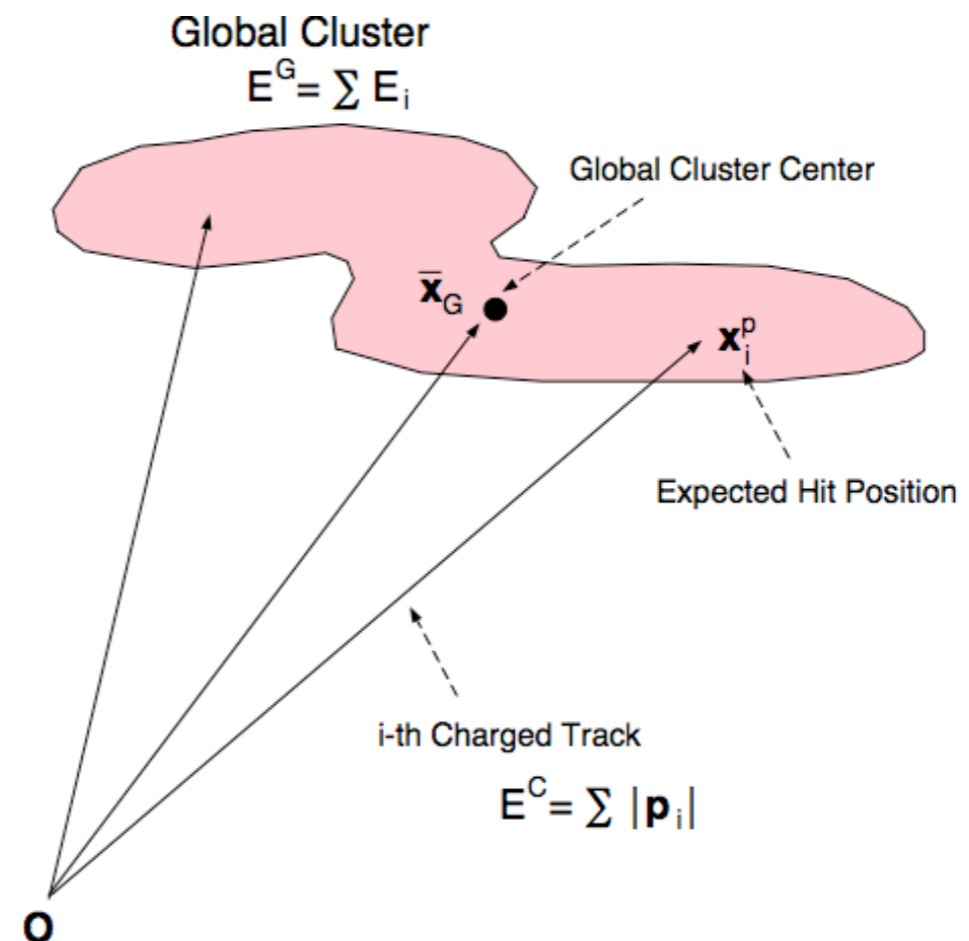


Figure 1.1: Global cluster and corresponding tracks

LEDA: Library Extension for Data Analysis

- Useful classes featuring ROOT capabilities

ANL2DSpline.h	KalTrackDim.h	TFFT.h	TObjInt.h	TVKalState.h
ANL2DVector.h	TAttDrawable.h	TH1E.h	TObjNum.h	TVKalSystem.h
ANL3DVector.h	TAttElement.h	THelicalTrack.h	TPlane.h	TVMeasLayer.h
ANL4DVector.h	TAttLockable.h	THype.h	TStraightTrack.h	TVSolid.h
ANLEventShape.h	TCircle.h	TKalDetCradle.h	TTube.h	TVSurface.h
ANLGaussFitter.h	TCylinder.h	TKalMatrix.h	TVAddress.h	TVTrack.h
ANLJetFinder.h	TDim.h	TKalTrack.h	TVCurve.h	TVTrackHit.h
ANLPairCombiner.h	TDynArray2.h	TKalTrackSite.h	TVKalDetector.h	
Anlib.h	TDynArray3.h	TKalTrackState.h	TVKalSite.h	

- Kalman filter classes => [details in Keisuke's talk](#)
- TAttLockable class adds lockable attribute to an object
- TAttDrawable class adds drawable attribute to an object
- ANL4DVector: lockable Lorentz vector class
- ANLPairCombiner: class library for pair combiners
- ANLEventShape: calculate Thrust, Oblateness, Major/Minor Axis

ttbar 6-jets pairing example

```
ANLJadeEJetFinder jclust(fYcut);
jclust.Initialize(tracks);
jclust.FindJets();
jclust.ForceNJets(xNjets);
TObjArray &jets = jclust.GetJets();

// Find W and top candidates in given mass windows.

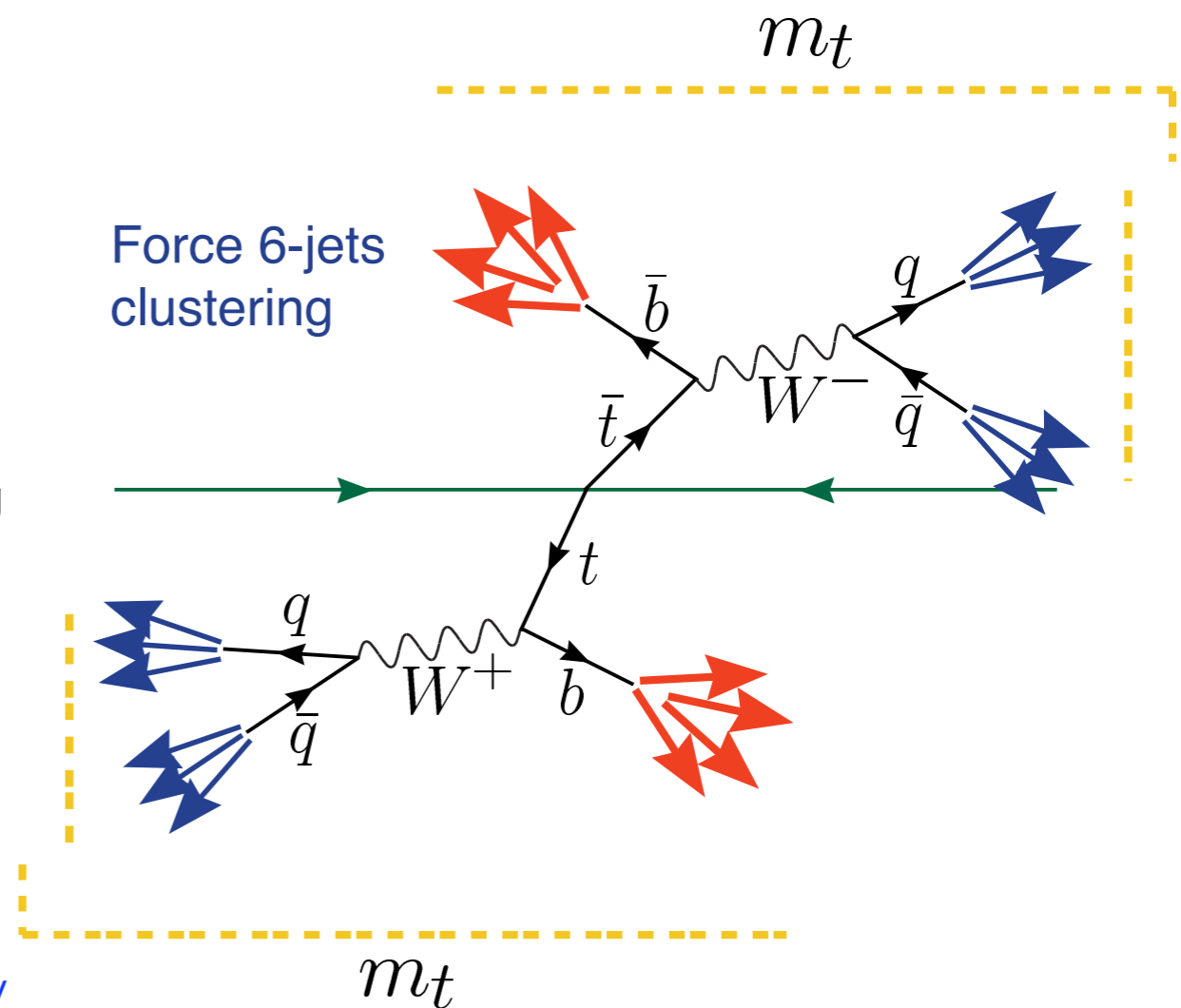
TObjArray solutions(90);
solutions.SetOwner();
ANLPairCombiner w1candidates(jets,jets);
ANLPair *w1p, *w2p, *bbp;
while ((w1p = static_cast<ANLPair *>(w1candidates()))) {
    ANLPair &w1 = *w1p;
    Double_t w1mass = w1().GetMass();
    if (TMath::Abs(w1mass - kMassW) > xM2j) continue;
    w1.LockChildren();

    ANLPairCombiner w2candidates(w1candidates);
    while ((w2p = static_cast<ANLPair *>(w2candidates()))) {
        ANLPair &w2 = *w2p;
        if (w2.IsLocked()) continue;
        Double_t w2mass = w2().GetMass();
        if (TMath::Abs(w2mass - kMassW) > xM2j) continue;
        w2.LockChildren();
```

```
        ANLPairCombiner bbcandidates(w2candidates);
        bbcandidates.Reset();
        while ((bbp = static_cast<ANLPair *>(bbcandidates()))) {
            ANLPair &bb = *bbp;
            if (bb.IsLocked()) continue;
            for (Int_t i = 0; i < 2; i++) {
                ANL4DVector *b1p = static_cast<ANL4DVector *>(bb[i]);
                ANL4DVector *b2p = static_cast<ANL4DVector *>(bb[1-i]);
                ANLPair *bw1p = new ANLPair(b1p,w1p);
                ANLPair *bw2p = new ANLPair(b2p,w2p);
                ANLPair &bw1 = *bw1p;
                ANLPair &bw2 = *bw2p;
                Double_t t1mass = bw1().GetMass();
                Double_t t2mass = bw2().GetMass();
                if (TMath::Abs(t1mass - kMasst) > xM3j ||
                    TMath::Abs(t2mass - kMasst) > xM3j) {
                    delete bw1p;
                    delete bw2p;
                    continue;
                }
                Double_t chi2 = TMath::Power((w1mass - kMassW)/kSigmaMw,2.)
                    + TMath::Power((w2mass - kMassW)/kSigmaMw,2.)
                    + TMath::Power((t1mass - kMasst)/kSigmaMt,2.)
                    + TMath::Power((t2mass - kMasst)/kSigmaMt,2.);
                solutions.Add(new ANLPair(bw1p,bw2p,chi2));
            }
        }
        w2.UnlockChildren();
    }
    w1.UnlockChildren();
}
```

ttbar -> 6jets reconstruction

- I) Force 6-jets clustering
- II) Confirm $\text{Max_cos } \theta_{\text{jet}}$ should be less than 0.99
- III) Choose all the 15-possible pairs out of 6-jets => W_1 candidate
- IV) Choose all the 6-possible paris out of remaining 4-jets => W_2 candidate
- V) **Remaining 2-jets should be b-jets**: flavor tagging (charm/bottom tagging) is very important to eliminate both combinatorial and process BGs
- VI) There are **2 possibilities to attach a b-jet to W_1 and W_2 candidates**
- VII) Store all solutions w/ $\chi^2 = (m_{w1} - m_w)^2 / \sigma_{mw}^2 + (m_{w2} - m_w)^2 / \sigma_{mw}^2 + (m_{t1} - m_t)^2 / \sigma_{mt}^2 + (m_{t2} - m_t)^2 / \sigma_{mt}^2$
- VIII) Sort solutions according to χ^2 : **choose the best solution**



M@RS

An ILD-DST analysis package

★ M@RS = Modular Analysis with Root-based Subprograms

❖ Aim

- ▶ provides a common framework for ILD-DST analysis (interface to LCIO) for JSF users
- ▶ same approach (= minimum user code modification) between Full simulator-Standard reconstruction (MarlinReco/PandoraPFA/LCFIVertex) and Quick simulator analyses
 - ➔ make maximum use of the past resources!!
- ▶ use ROOT and OO features maximally = efficient reconstruction for complicated final states (e.g. ttbar -> 6jets) using LEDA/Anlib; ANL4DVector (Lockable TLorentzVector), ANLPair etc.

❖ Dependencies

- ▶ ROOT, LEDA/Anlib and JSF-kern

❖ Provides

- ▶ [MarsPFObject](#): `LCCollection *colPFOPtr = gLCEventPtr->getCollection("PandoraPFOs");`
- ▶ [MarsJet](#): `LCCollection *colJetPtr = gLCEventPtr->getCollection("Durham_6Jets");`

Summary

- In Asia (mainly in Japan), most of physics feasibility studies and ILD physics benchmarking are done and on-going by using JSF (with an ILD-DST interface or a fast detector simulator, taking advantage of many ROOT features)
- Requests from JSF users
 - Want to keep our momentum using “ILD software”
 - Fast detector simulator in the Marlin framework
 - Common analyses framework for both FullSim and FastSim data structure
 - LEDA in the Marlin framework