Nuisance Check Update

1 Introduction

The nuisance check package has been updated to allow iterative fixing of NPs during fitting. This is done to check the effect NP removal has on each fit. This has been implemented as a new algorithm called npRemovedCompare.
The algorithm works by specifying three different types of NP.

- Free → NP is always free
- Fix → NP is always fixed
- Variable → An additional fit is done with this variable fixed. All other iterations will have it free.

The first fit frees all variable/free NP and holds constant the fixed NP. After this each variable NP is fixed for a single iteration of the fit and then freed up again. This process is repeated for all variable NP.

2 Installation and Running

The installation is the same as the NuisanceCheck code given here. However the code at the moment is stored in my public:
/afs/phas.gla.ac.uk/user/a/amorton/public/NuiCheck/FitCrossCheckForLimits.C
Code can be ran as a dynamic library executable, executable function or by python scripts. This new function has been added to the first two but not the python scripts.

3 Inputs

The inputs for this algorithm is the same as any other function in nuisance check

```
void PlotFitCrossChecks(const char* infile          = "WorkspaceForTest1.
    root",
                const char* outputdir       = "./results/",
                const char* workspaceName   = "combined",
                const char* modelConfigName = "ModelConfig",
                const char* ObsDataName     = "obsData"){
////...
}
```

Figure 1: The inputs for any algorithm of nuisance check. Note these all depend on the WS.

In addition to this other parameters must be set to use the NPRemover function. The main issue is specifying which type of variable (free/fix/variable) each NP should be set. At the moment this is done by specifying three variables

- nuiFree → vector of NP names
- nuiFix → vector of NP names
- freeAll → boolean

A NP can fall into the following categories

- In both lists → The NP is variable
- In one of the lists → The NP is either fixed/floated depending on list which it is included
- Is not listed and freeAll=true → The NP is made free
- Is not listed and freeAll=false → The NP is made variable

```
//
// 0 - plot chi2/mu/pulls after a set or single NP has been removed.
//

static const string fixArr[] = { "alpha_MUONS_SCALE", "
alpha_MODEL_PAR_TTbar_aMcAtNlo", "alpha_Luminosity"  };
vector<string> nuiFix (fixArr, fixArr + sizeof(fixArr) / sizeof(fixArr
[0]) );
static const string freeArr[] = { "alpha_MUONS_SCALE", "
alpha_MODEL_PAR_TTbar_aMcAtNlo", "alpha_Luminosity" };
vector<string> nuiFree (freeArr, freeArr + sizeof(freeArr) / sizeof(
freeArr[0]) );
string name = "0lep_3TeV_AllNP_bin100";
bool freeAll=true; ///If the NP is not listed then it is assumed to be
free if freeAll=true OR if freeAll=false then all not included are
assumed to fix during a fit.
npRemovedCompare(nuiFix,nuiFree,freeAll, IsConditional , 0.0);
```

Figure 2: Input for function. Two list (nuiFree,nuiFix) and bool (freeAll) are shown.

This is a slightly silly way to specify variables but this is currently what is implemented.

4 outputs

The results of the fit are then added to the TFile shared by all nuisance check algorithms. The algorithm stores histograms and graphs, with canvases created if needed. To store the canvases the global variable createCanvas must be true.

- Chi2
- Mu
- Mu (constraint order)
- Pull for each NP
- Pull (Constraint order) for each NP

Figure 3: List of output histograms/graphs

Furthermore a comparison of NP after fixing is made and printed at the end of the running. This includes as output

- Max/Min minNll
- Max/Min Mu
- All NP after NP removed over a certain limit
- All constraints after NP removed over a certain limit

0.1 Introduction

This is designed to be an introduction to any analysis and will cover some of the basic ideas needed. However a focus on the the VH analysis give here

0.2 Theory and past searches

Need to be filled in after going through theory course again.

0.3 Data and MC samples

Pileup must be taken into account. This is done via reweighting. The quality of the data is validated using the GRL and DQ flags. The simulation of signal and background can be viewed in **??**.
The k-factor is a factor which relates the leading order to the error compared to the full calculation. The filter term is to relate the number of events expected to the number produced.

To begin to discuss physics objects you need to begin at the xAOD root files created by simulation and data. You can see all the collections and type in an xAOD using:
checkxAOD.py
The ATLAS Muon Combined Performance Group define muons using a series of cut points, this is done for inner detector and muon spectrometer. Both the ID and MS are used in combination to determine the pt and mass of the muon. Note the pt and resolution depend on many factors (magnetic field, scattering, intrinsic resolution), therefore momentum scale and resolution corrections are also applied. This makes the MC and data follow the same distribution. These were determined using the J/ψ ($c\bar{c}$)/Z resonance. Reconstruction efficiency is allowed to vary with a scale factor which is the ratio of MC/data. This if course depends of pt and η.
Isolation criteria for leptons is needed since semi leptonic/hadronic decays are the main source of non-prompt/misidentified leptons. A tight cut is performed on each lepton by constructing the energy for the track/Calo object inside a R=0.2 cone and then doing a cut which depends on p_t and η. The energy must be over this cut value. A variable R cut is also performed on tracks only for the loosetrackonly requirement. This reduces the size of the cone since the Z $\rightarrow$ ll cone will reduce in size for higher pt.
Calorimeter jets are corrected by adding muon inside jet to jet. $H \rightarrow b\bar{b}$ reconstructed with 1.0 large R jet, which focuses on pt ¿ 250 GeV since less than this will be in a larger jet. Smaller R jets are used for Etmiss which must use a set eta and pt ¿ 20 GeV for JES calibration to be valid. Note the JES must be compared to MC since the real detector will have some energy loses. This is also an additional uncertainty for the final fit. To reduce jets not from the primary vertices other than hard scattering vertex JVT recommendations for pt ¡ 50 GeV smaller R jets is used. JVT is used to reduce pile up by determining the primary vertex and then determining the amount of energy which comes from this.
Overlapping jets with electrons are removed if within R ¡ 0.2

0.4 Triggers

The trigger efficiency depends on a particular sample. A sample here is a selection of cuts after requiring these triggers. The efficiency is determined by taking a selection of events from the sample and running this through the trigger. The efficiency is just the ratio before and after the trigger. Note the efficiency will depend of some observable, e.g VH mass. A scale factor must also be used to determine the different between MC and data.

0.5 Selection

The number of VHLoose lepton number is varied to keep the 0/1/2 lepton channel orthogonal.
In the two lepton channel the muon momentum resolution deteriorates at high pt so the pt is scaled by $\frac{m_{ZZ}}{m_{\mu\mu}}$.

W mass constraint used in the 1 lepton channel to determine neutrino pt.
Geometric cuts are used in the 0 lepton channel.
After these kinematic cuts then the H $\rightarrow b\bar{b}$ is exploited to increase sensitivity. This is also used to define control regions. This is done looking at the mass and number of b tags.

0.6 Multijet

0.7 Nuisance Check update (NP Pruning)

0.8 CMake Introduction)

0.9 CxAODMaker

0.10 Abbreviations

- EDM $\rightarrow$ Event Data Model
- GRL $\rightarrow$ Good Run List
- JES $\rightarrow$ Jet Energy Scale
- JVT *rightarrow* Jet Vertex Tagger