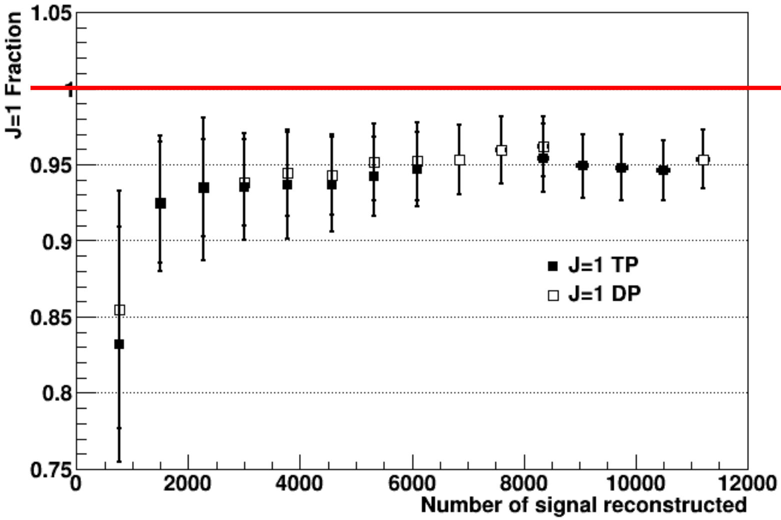
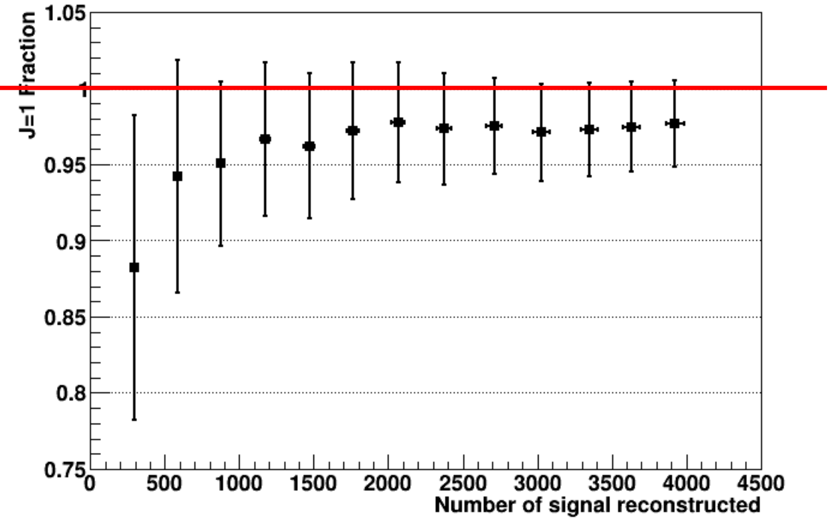
$K^+K^-\pi^0$ update

Fraction of J=0 events identified versus number of reconstructed signal events

$a_0\pi^0$



$(K^+K^-)_S\pi^0$



- Thrown-acceptance events are 1 million phase space events
- Statistical errors are consistent with nominal value 1.0 when the distribution of mass[K^+K^-] of thrown-acceptance match the distribution of thrown-signal

As a **TEST**: Shaping thrown acceptance events

As a **TEST**: Shaping thrown acceptance events

- The procedure in this **TEST** is not practical way to perform an analysis.

As a **TEST**: Shaping thrown acceptance events

- The procedure in this **TEST** is not practical way to perform an analysis. It is simply a **TEST.**

As a **TEST**: Shaping thrown acceptance events

- The procedure in this **TEST** is not practical way to perform an analysis. It is simply a **TEST.**

- Shaped the a_0 isobar for the acceptance events exactly as signal.

As a **TEST**: Shaping thrown acceptance events

- The procedure in this **TEST** is not practical way to perform an analysis. It is simply a **TEST.**

- Shaped the a_0 isobar for the acceptance events exactly as signal. NOTE: Once the a_0 isobar is in the acceptance events, there is no need for that shape to exist in the PWA

ASU

As a **TEST**: Shaping thrown acceptance events

- The procedure in this **TEST** is not practical way to perform an analysis. It is simply a **TEST.**

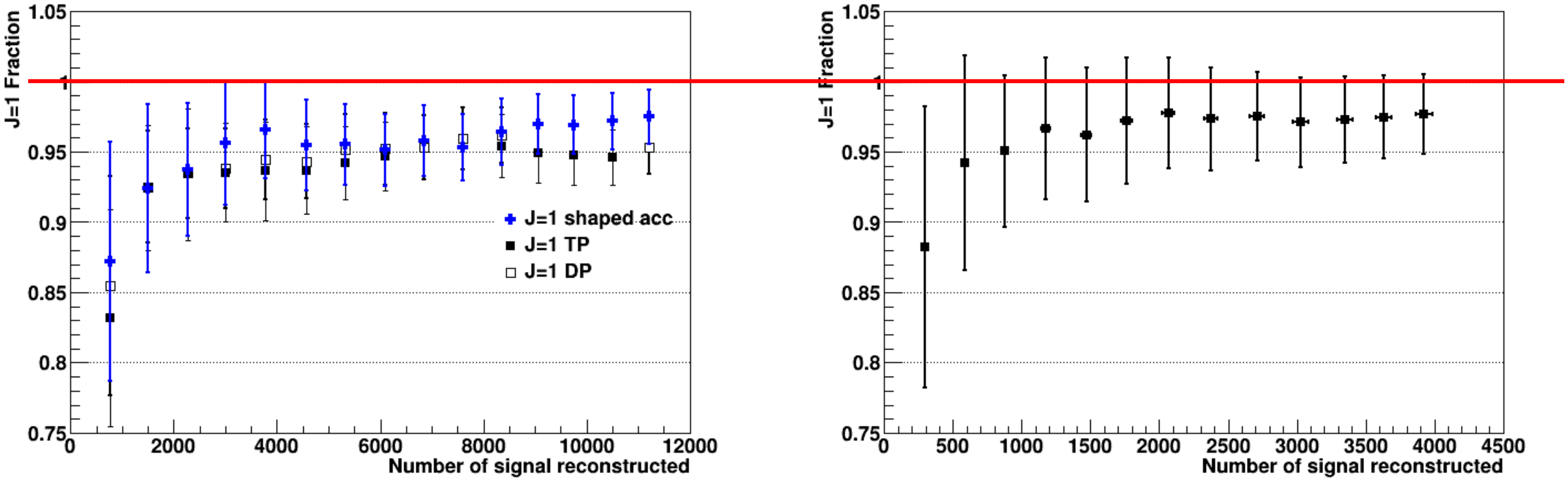
- Shaped the a_0 isobar for the acceptance events exactly as signal. NOTE: Once the a_0 isobar is in the acceptance events, there is no need for that shape to exist in the PWA

- Removed a_0 shape from PWA, but kept phase by dividing complex Flatte function by the magnitude of Flatte function

ASU

Fraction of J=0 events identified versus number of reconstructed signal events

$a_0\pi^0$ $(K^+K^-)_S\pi^0$



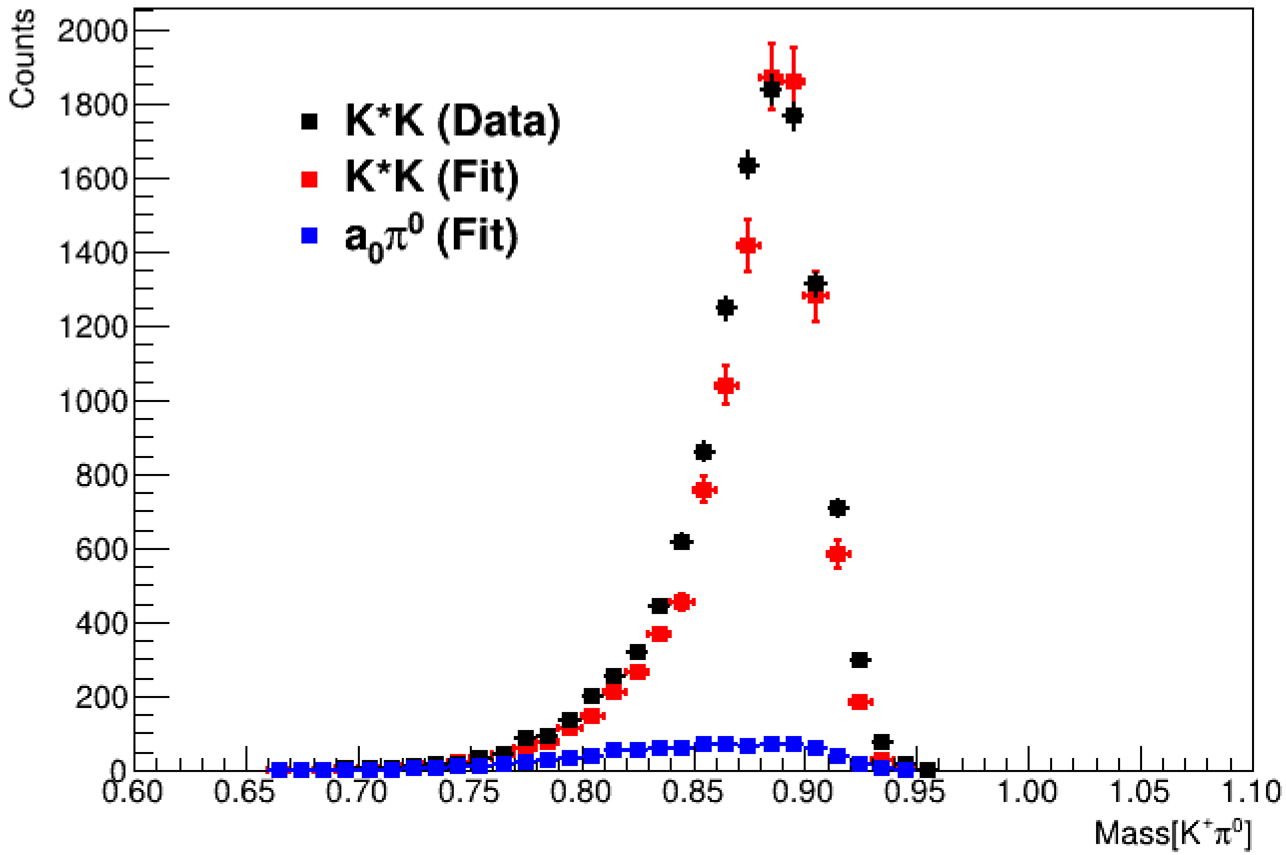
- Thrown-acceptance events are 1 million phase space events

- The PWA results of the isobar shaped acceptance $a_0\pi^0$ events are more consistent with the PWA from the $(K^+K^-)_s\pi^0$ events

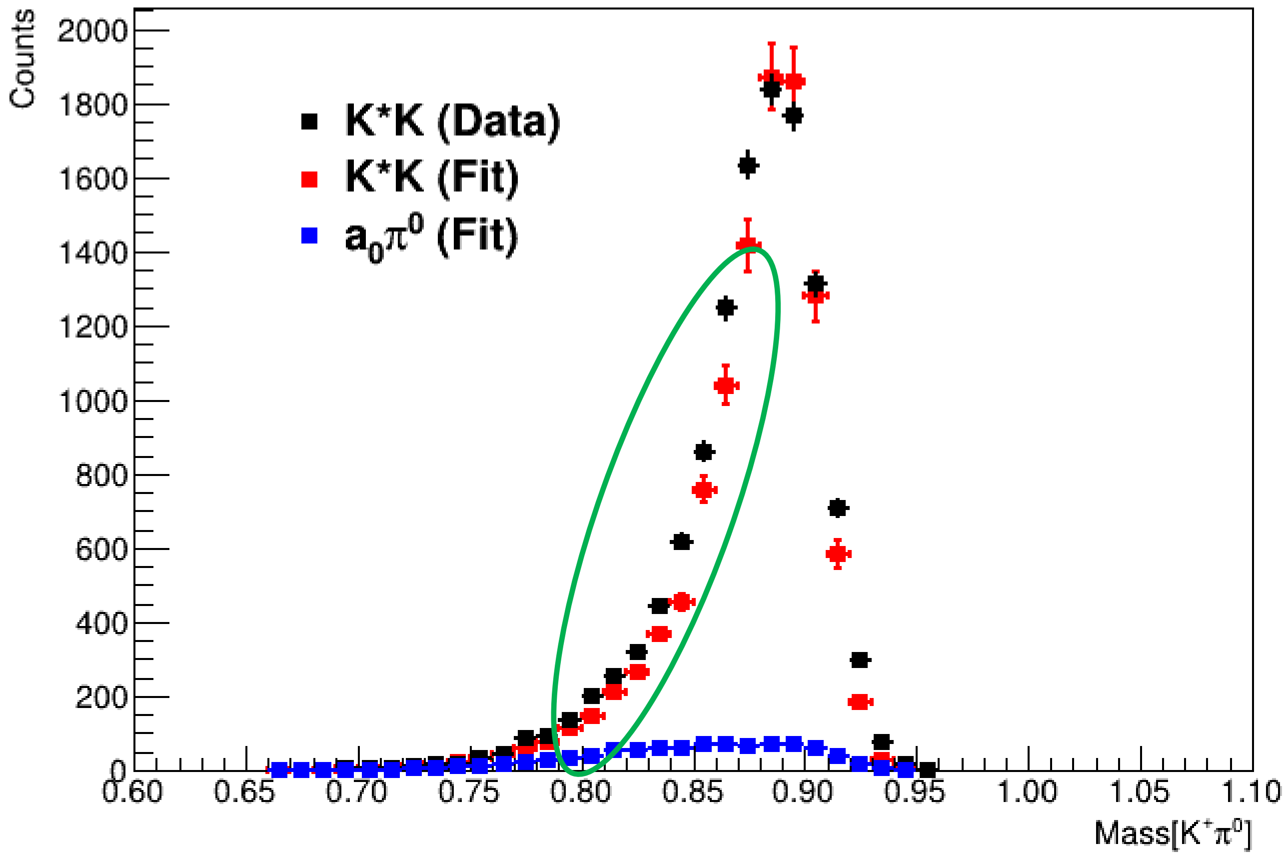
K^* isobar fits

- Generated $K^{+*}K$, where $K^{+*} \rightarrow K^+\pi^0$ and J=1, l=0, s=1
- Fit mass[$K^+\pi^0$] with K^*(892) and also a_0

K^* isobar fits

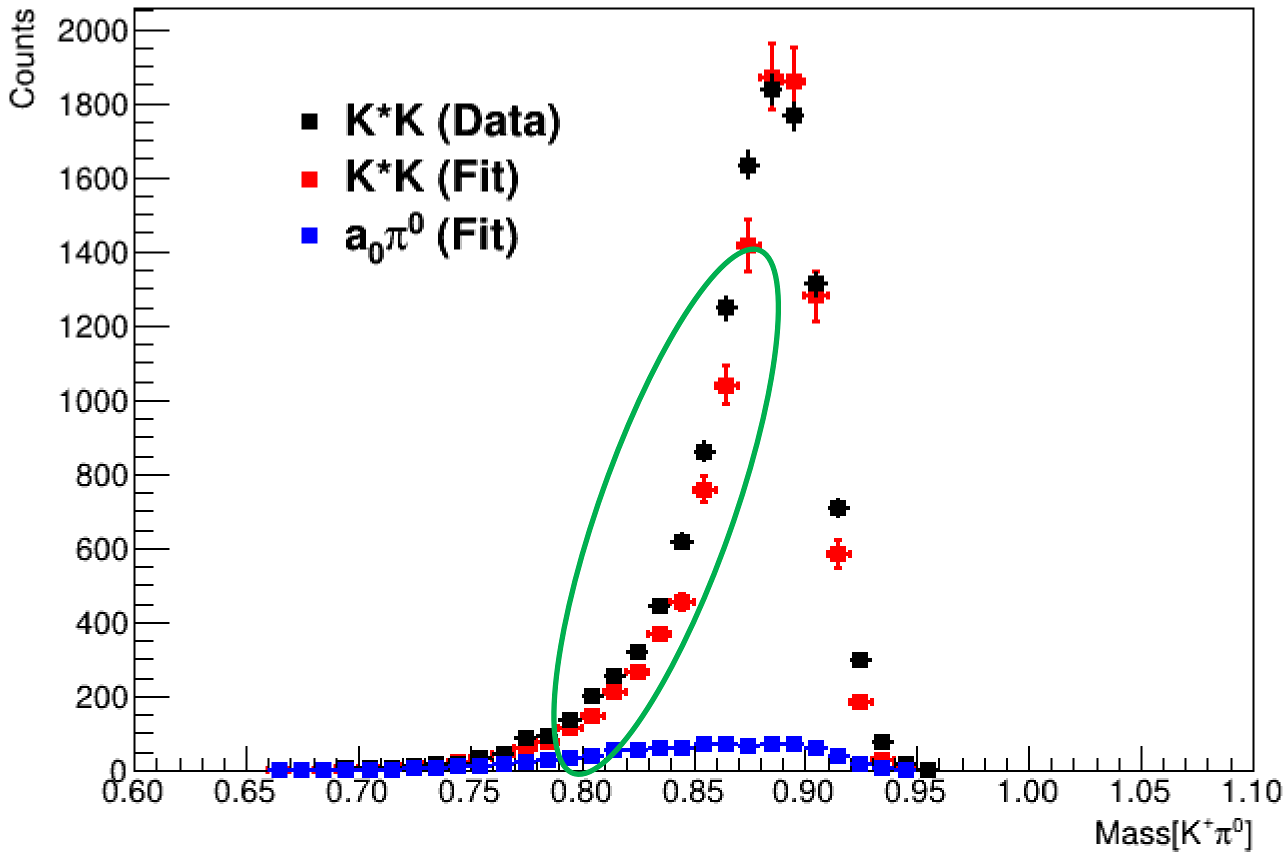


K^* isobar fits



- Reconstructed data has been smeared by detector

K^* isobar fits



- Reconstructed data has been smeared by detector
- Must put smear into the fitted Breit-Wigner

K^* isobar smear fits

- Modified standard AmpTools Breit-Wigner by numerical convolution with a gaussian. See backup slides for more details.

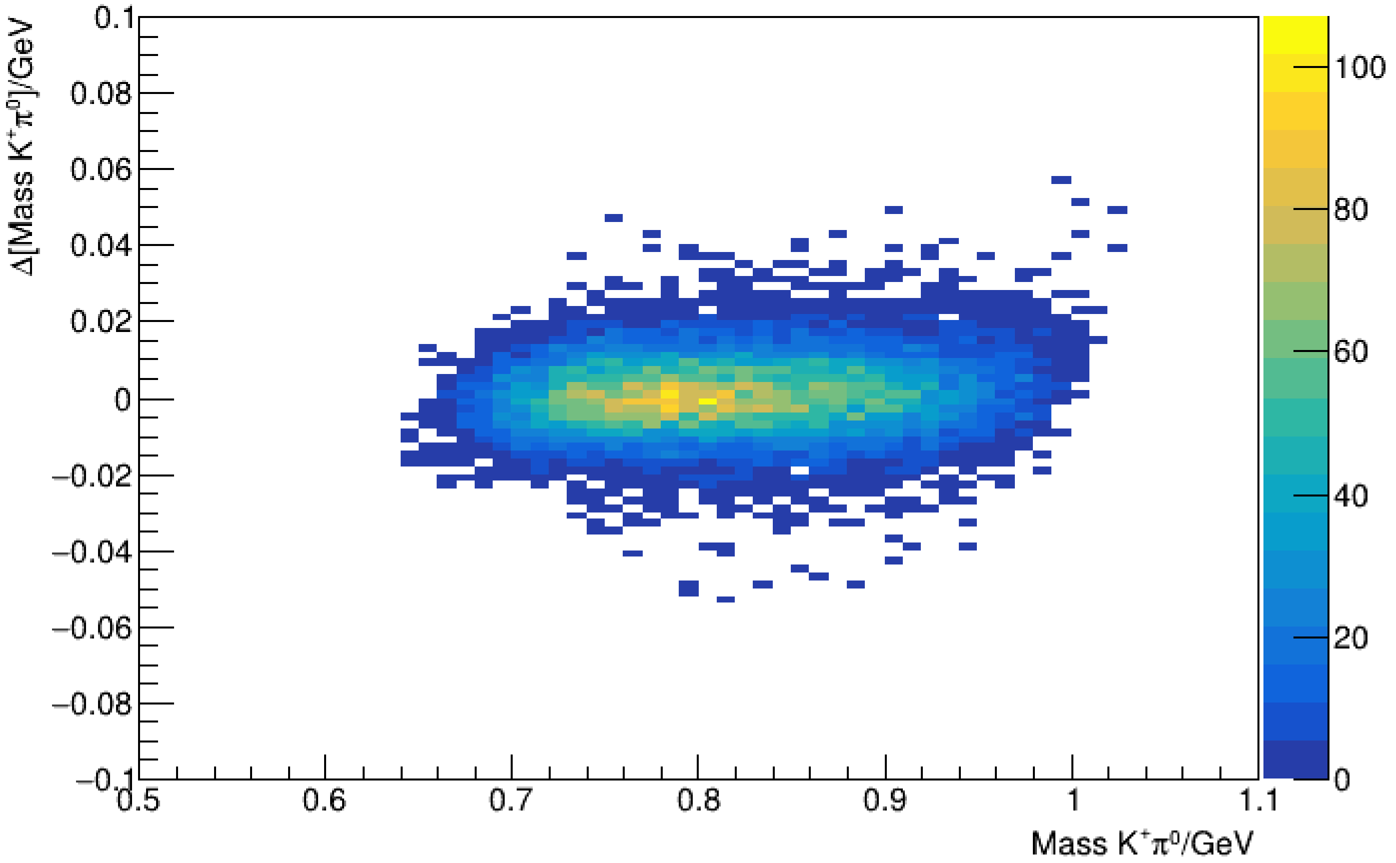
K^* isobar smear fits

- Modified standard AmpTools Breit-Wigner by numerical convolution with a gaussian. See backup slides for more details.

- The standard deviation of the gaussian was determined by looking at

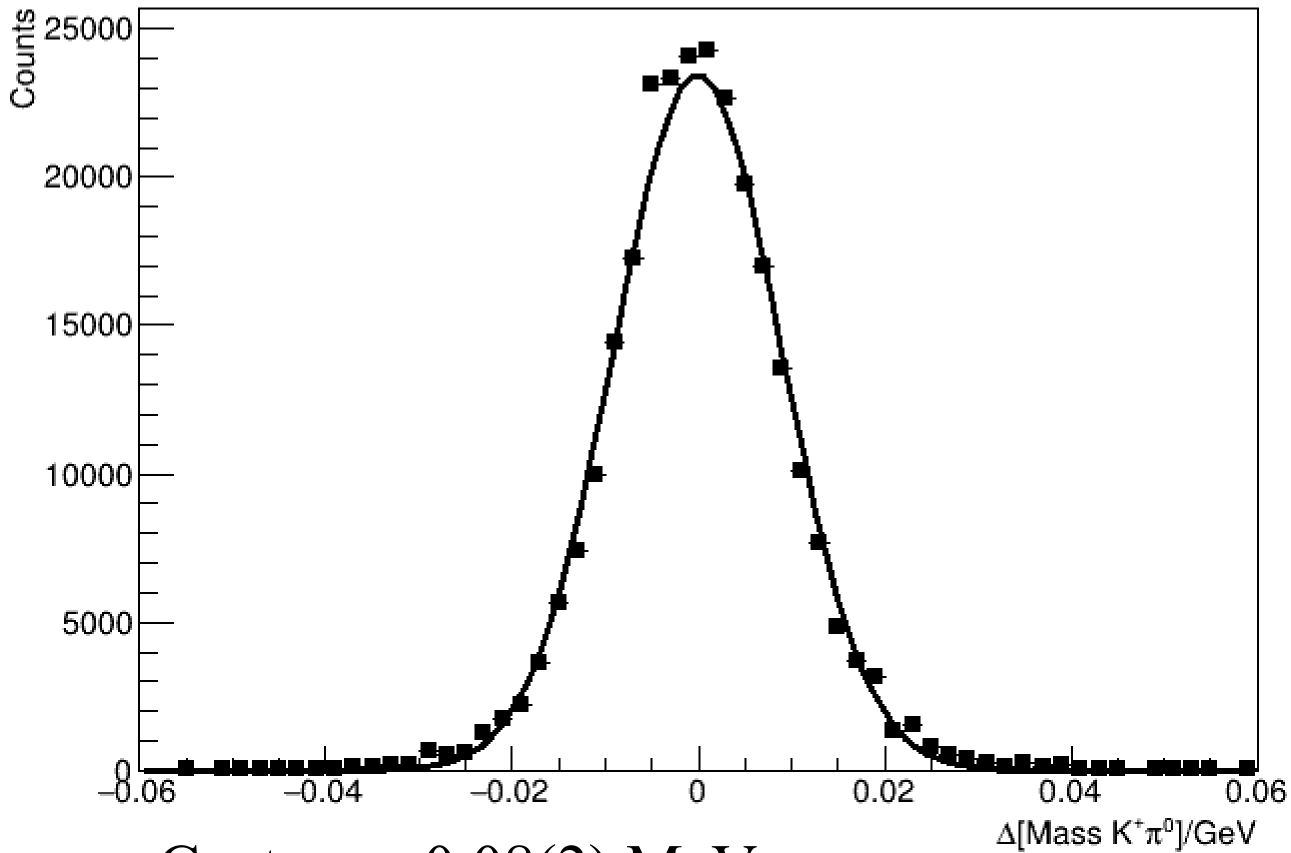
Δmass[$K^+\pi^0$] = (mass detector precision) – (mass thrown precision)

$\Delta[\text{Mass}(K^+\pi^0)]$ versus $\text{Mass}(K^+\pi^0)$



- From phase space distribution

$\Delta[\mathrm{Mass}(K^+\pi^0)]$ fit to gaussian

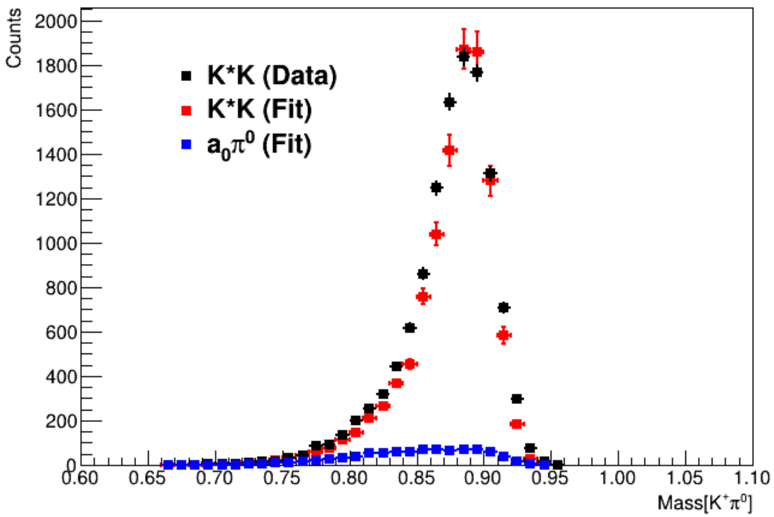


Center = -0.08(2) MeV

σ = 9.01(2) MeV

K^* isobar fits (no angular information fit)

Without gaussian convolution

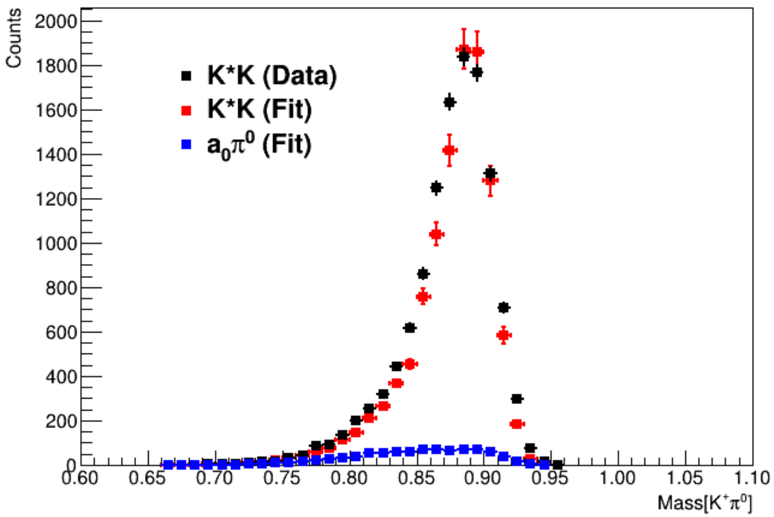


Fraction $K^*K = 0.92(3)$

Fraction $a_0\pi^0 = 0.07(1)$

K^* isobar fits (no angular information fit)

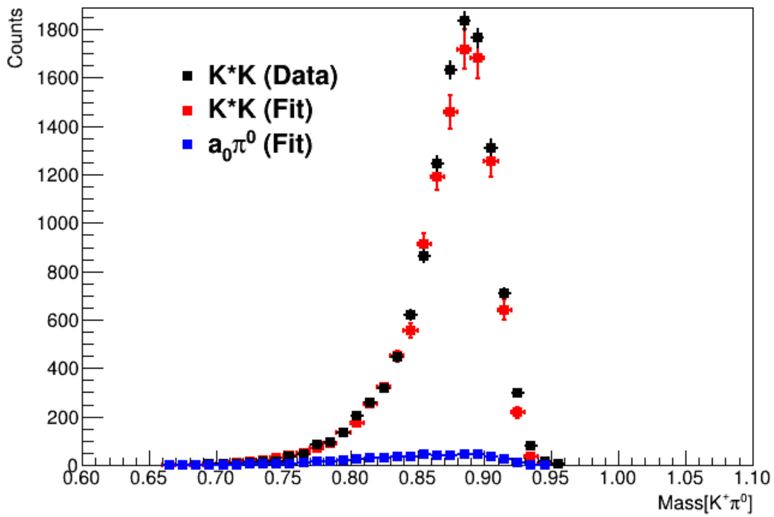
Without gaussian convolution



Fraction $K^*K = 0.92(3)$

Fraction $a_0\pi^0 = 0.07(1)$

With gaussian convolution



Fraction $K^*K = 0.98(3)$

Fraction $a_0\pi^0 = 0.04(1)$

ASU

PWA fits to $K^{*+}K^{-}$ including angular information

- Generated $K^{*+}K^{-}$ with j=1, l=0, s=1
-

PWA fits to $K^{*+}K^{-}$ including angular information

- Generated $K^{*+}K^{-}$ with j=1, l=0, s=1
- Fit designations
  - J=0 includes
    -

PWA fits to $K^{*+}K^{-}$ including angular information

- Generated $K^{*+}K^{-}$ with j=1, l=0, s=1
- Fit designations
  - J=0 includes
    - $a^0\pi^0$ with j=0, l=0, s=0
    - $K^{*+}K^{-}$ with j=0, l=1, s=1
    - $K^{*-}K^{+}$ with j=0, l=1, s=1
  -

PWA fits to $K^{*+}K^{-}$ including angular information

- Generated $K^{*+}K^{-}$ with j=1, l=0, s=1
- Fit designations
  - J=0 includes
    - $a^0\pi^0$ with j=0, l=0, s=0
    - $K^{*+}K^{-}$ with j=0, l=1, s=1
    - $K^{*-}K^{+}$ with j=0, l=1, s=1
  - J=1 includes
    - $a^0\pi^0$ with j=1, l=1, s=0
    - $K^{*+}K^{-}$ with j=1, l=0, s=1
    - $K^{*-}K^{+}$ with j=1, l=0, s=1
    - $K^{*+}K^{-}$ with j=1, l=1, s=1
    - $K^{*-}K^{+}$ with j=1, l=1, s=1
  -

ASU

PWA fits to $K^{*+}K^{-}$ including angular information

- Generated $K^{*+}K^{-}$ with j=1, l=0, s=1
- Fit designations
  - J=0 includes
    - $a^0\pi^0$ with j=0, l=0, s=0
    - $K^{*+}K^{-}$ with j=0, l=1, s=1
    - $K^{*-}K^{+}$ with j=0, l=1, s=1
  - J=1 includes
    - $a^0\pi^0$ with j=1, l=1, s=0
    - $K^{*+}K^{-}$ with j=1, l=0, s=1
    - $K^{*-}K^{+}$ with j=1, l=0, s=1
    - $K^{*+}K^{-}$ with j=1, l=1, s=1
    - $K^{*-}K^{+}$ with j=1, l=1, s=1
  - J=2 includes
    - $K^{*+}K^{-}$ with j=1, l=2, s=1
    - $K^{*-}K^{+}$ with j=1, l=2, s=1

ASU

PWA fits to $K^{*+}K^{-}$ including angular information

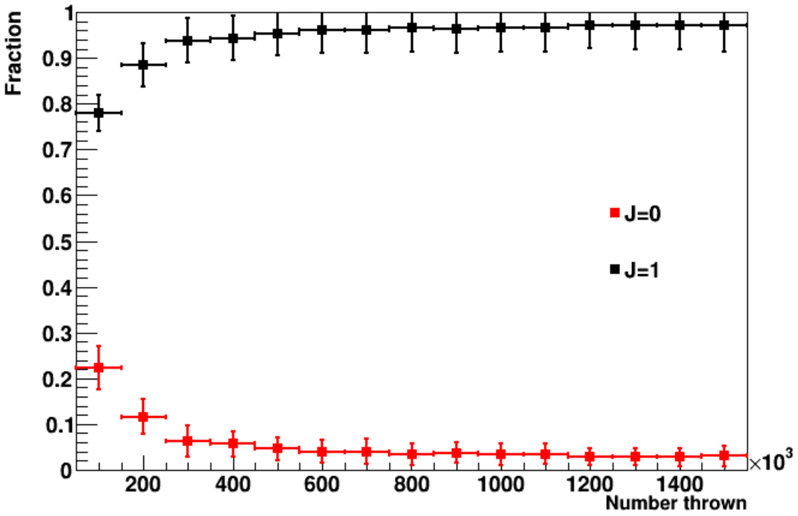
- Generated $K^{*+}K^{-}$ with j=1, l=0, s=1
- Fit designations
  - J=0 includes
    - $a^0\pi^0$ with j=0, l=0, s=0
    - $K^{*+}K^{-}$ with j=0, l=1, s=1
    - $K^{*-}K^{+}$ with j=0, l=1, s=1
  - J=1 includes
    - $a^0\pi^0$ with j=1, l=1, s=0
    - $K^{*+}K^{-}$ with j=1, l=0, s=1
    - $K^{*-}K^{+}$ with j=1, l=0, s=1
    - $K^{*+}K^{-}$ with j=1, l=1, s=1
    - $K^{*-}K^{+}$ with j=1, l=1, s=1
  - J=2 includes
    - $K^{*+}K^{-}$ with j=1, l=2, s=1
    - $K^{*-}K^{+}$ with j=1, l=2, s=1

→ **Same choice as E852**

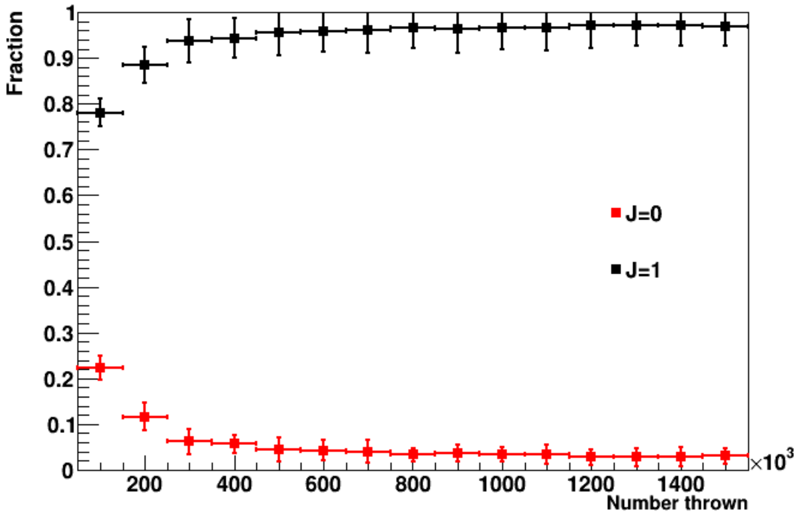
ASU

Fraction of J=0, 1 events identified

Without gaussian convolution



With gaussian convolution

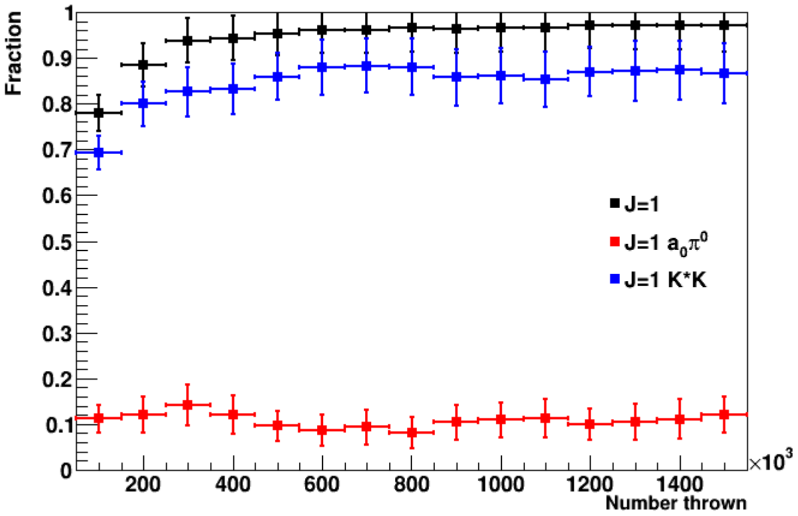


- 200 thousand thrown signal events with mass[$K^+K^-\pi^0$] = 1.415 GeV
- Varied the number of acceptance events thrown
- PWA fit includes J=0 and J=1
- The gaussian convolution does not help (or hurt) in resolving J

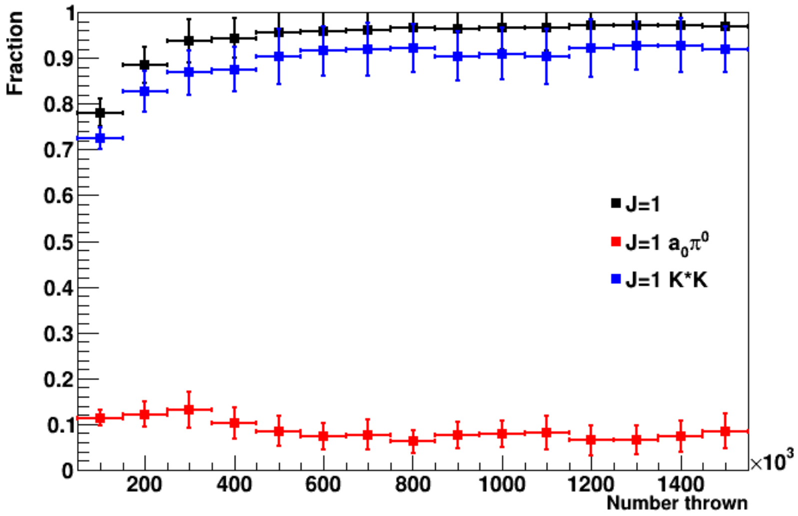
ASU

Fraction of $J = 1$ events identified by decay

Without gaussian convolution



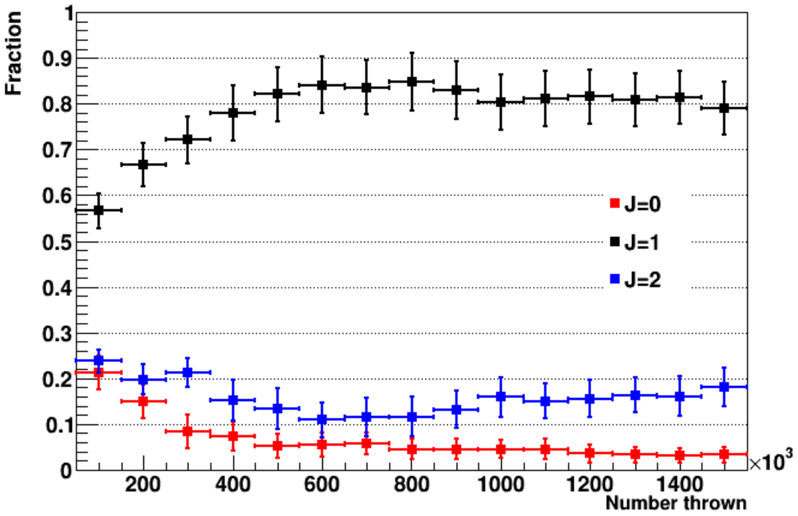
With gaussian convolution



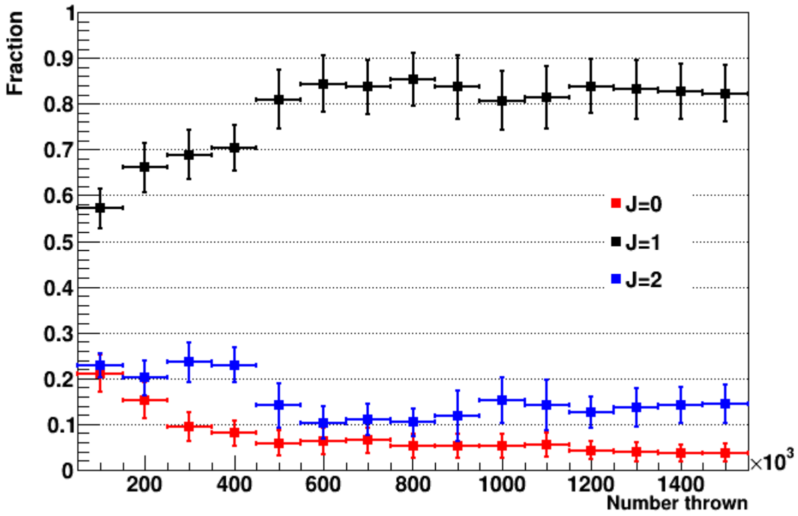
- 200 thousand thrown signal events with mass[$K^+K^-\pi^0$] = 1.415 GeV
- Varied the number of acceptance events thrown
- PWA fit includes J=0 and J=1
- The gaussian convolution still helps in resolving Isobar type

Fraction of J=0, 1, 2 events identified

Without gaussian convolution



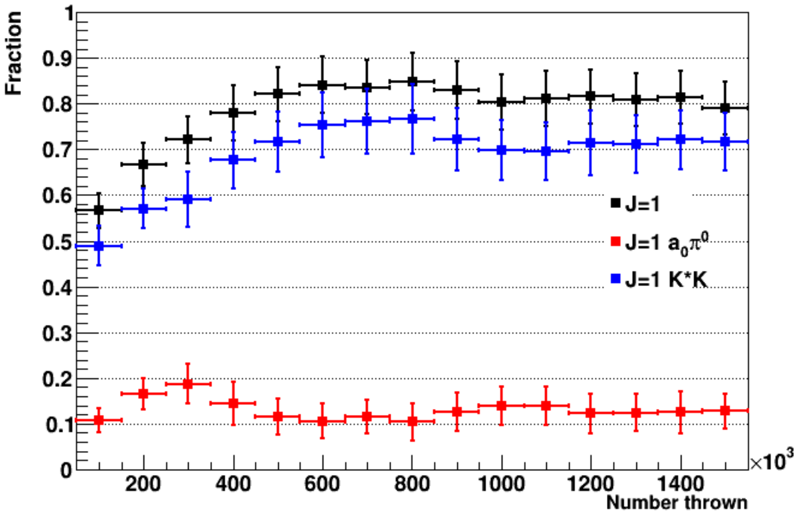
With gaussian convolution



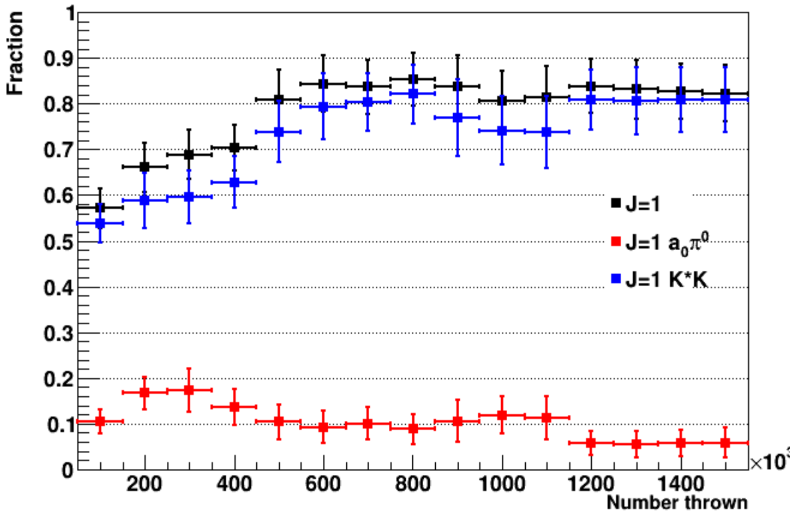
- 200 thousand thrown signal events with mass[$K^+K^-\pi^0$] = 1.415 GeV
- Varied the number of acceptance events thrown
- PWA fit includes J=0, J=1 and J=2
- The gaussian convolution helps slightly in resolving J

Fraction of J= 1events identified by decay

Without gaussian convolution



With gaussian convolution

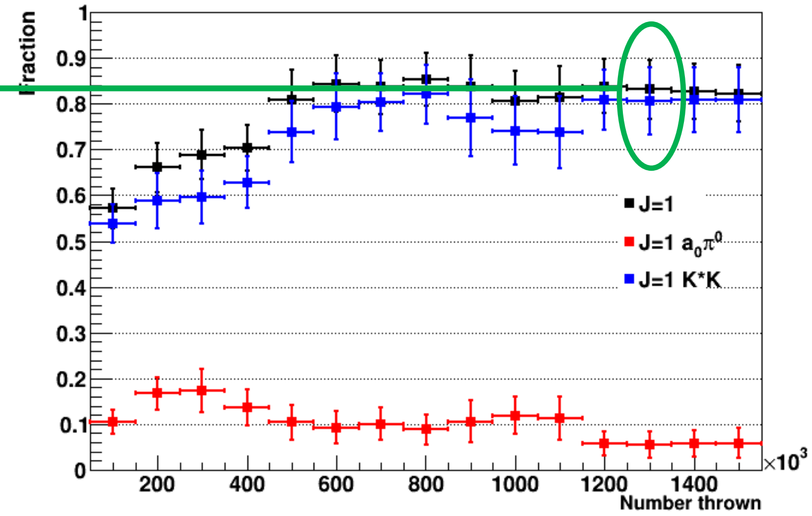


- 200 thousand thrown signal events with mass[$K^+K^-\pi^0$] = 1.415 GeV
- Varied the number of acceptance events thrown
- PWA fit includes J=0, J=1 and J=2
- The gaussian convolution still helps in resolving Isobar type

Fraction of $J = 1$ events identified by decay

With gaussian convolution

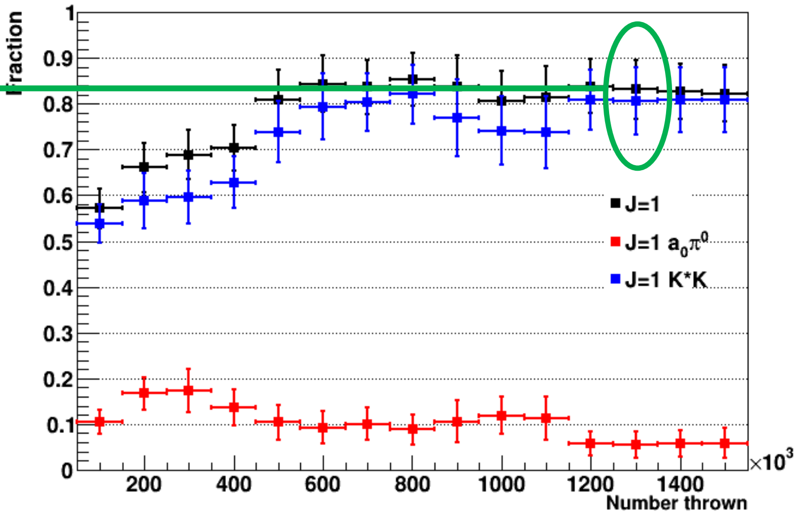
Looks stable for 1.3 million thrown acceptance events



Fraction of $J = 1$ events identified by decay

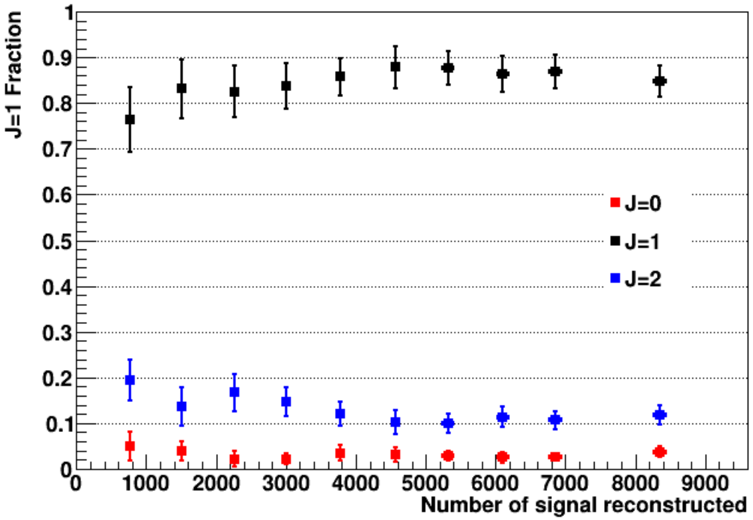
With gaussian convolution

Looks stable for 1.3 million thrown acceptance events



Will now use 1.3 million thrown acceptance events and vary the number of signal events

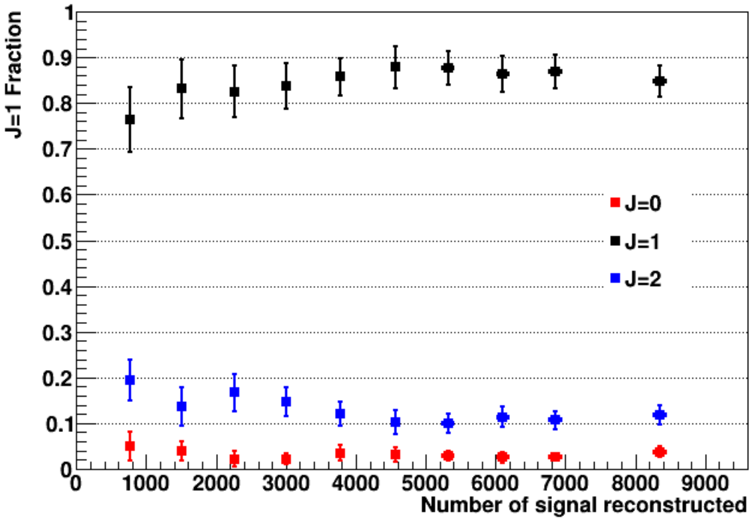
Fraction of events identified

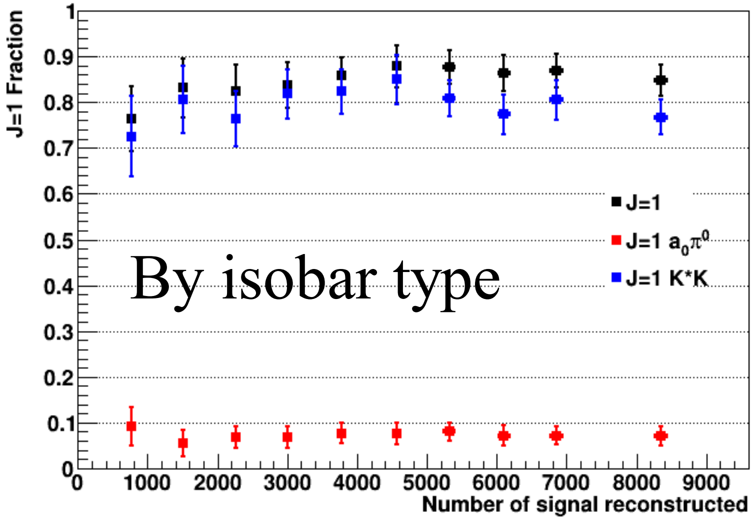


- 1.3 million thrown acceptance events with mass[$K^+K^-\pi^0$] = 1.415 GeV
- Varied the number reconstructed signal events
- PWA fit includes J=0, J=1 and J=2
- The gaussian convolution is now included as a default for K^*

ASU

Fraction of events identified





- 1.3 million thrown acceptance events with mass$[K^+K^-\pi^0]$ = 1.415 GeV
- Varied the number reconstructed signal events
- PWA fit includes J=0, J=1 and J=2
- The gaussian convolution is now included as a default for K^*

Title

Backup slides

Breit-Wigner with gaussian convolution

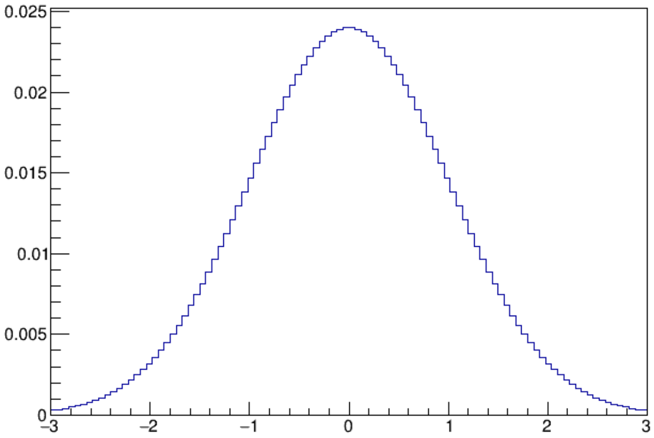
If interested, code can be found at:

https://www.public.asu.edu/~dugger/BWSmear/

Some details

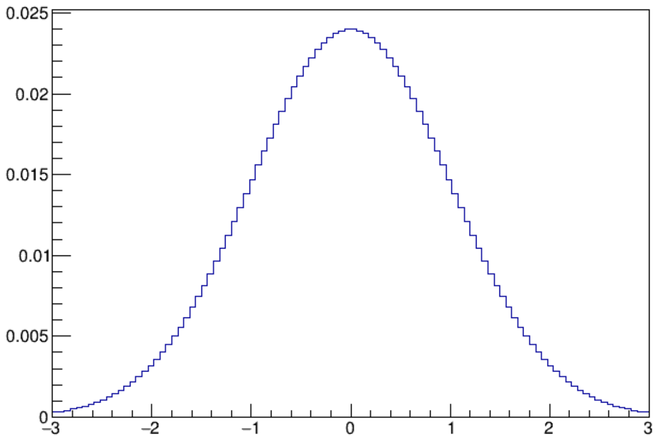
Gaussian convolution

- Created gaussian histogram with
  - 100 bins
  - Standard deviation set to one
  - Center set to zero
  - Normalized to one
- Pulled numbers out and put into header file



Gaussian convolution

- Created gaussian histogram with
  - 100 bins
  - Standard deviation set to one
  - Center set to zero
  - Normalized to one
- Pulled numbers out and put into header file

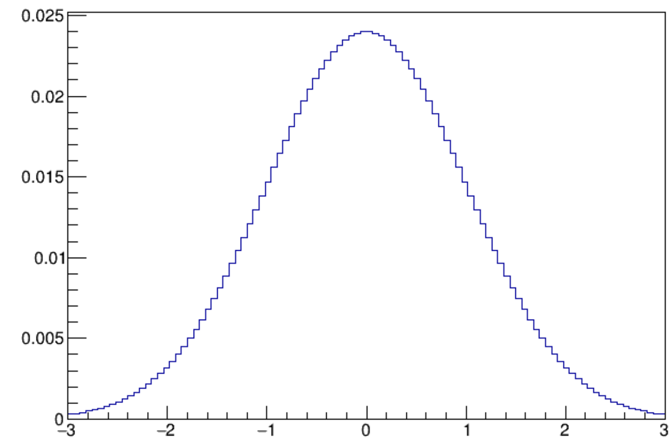



```
double gCenterArray[100] = {
  -2.97, -2.91, -2.85, -2.79, -2.73, -2.67, -2.61, -2.55, -2.49, -2.43,
  -2.37, -2.31, -2.25, -2.19, -2.13, -2.07, -2.01, -1.95, -1.89, -1.83,
  -1.77, -1.71, -1.65, -1.59, -1.53, -1.47, -1.41, -1.35, -1.29, -1.23,
  -1.17, -1.11, -1.05, -0.99, -0.93, -0.87, -0.81, -0.75, -0.69, -0.63,
  -0.57, -0.51, -0.45, -0.39, -0.33, -0.27, -0.21, -0.15, -0.09, -0.03,
  0.03, 0.09, 0.15, 0.21, 0.27, 0.33, 0.39, 0.45, 0.51, 0.57,
  0.63, 0.69, 0.75, 0.81, 0.87, 0.93, 0.99, 1.05, 1.11, 1.17,
  1.23, 1.29, 1.35, 1.41, 1.47, 1.53, 1.59, 1.65, 1.71, 1.77,
  1.83, 1.89, 1.95, 2.01, 2.07, 2.13, 2.19, 2.25, 2.31, 2.37,
  2.43, 2.49, 2.55, 2.61, 2.67, 2.73, 2.79, 2.85, 2.91, 2.97
};
```

→ **Bin centers**

Gaussian convolution

- Created gaussian histogram with
  - 100 bins
  - Standard deviation set to one
  - Center set to zero
  - Normalized to one
- Pulled numbers out and put into header file




```
double gCenterArray[100] = {
  -2.97, -2.91, -2.85, -2.79, -2.73, -2.67, -2.61, -2.55, -2.49, -2.43,
  -2.37, -2.31, -2.25, -2.19, -2.13, -2.07, -2.01, -1.95, -1.89, -1.83,
  -1.77, -1.71, -1.65, -1.59, -1.53, -1.47, -1.41, -1.35, -1.29, -1.23,
  -1.17, -1.11, -1.05, -0.99, -0.93, -0.87, -0.81, -0.75, -0.69, -0.63,
  -0.57, -0.51, -0.45, -0.39, -0.33, -0.27, -0.21, -0.15, -0.09, -0.03,
  0.03, 0.09, 0.15, 0.21, 0.27, 0.33, 0.39, 0.45, 0.51, 0.57,
  0.63, 0.69, 0.75, 0.81, 0.87, 0.93, 0.99, 1.05, 1.11, 1.17,
  1.23, 1.29, 1.35, 1.41, 1.47, 1.53, 1.59, 1.65, 1.71, 1.77,
  1.83, 1.89, 1.95, 2.01, 2.07, 2.13, 2.19, 2.25, 2.31, 2.37,
  2.43, 2.49, 2.55, 2.61, 2.67, 2.73, 2.79, 2.85, 2.91, 2.97
};
```

Bin contents

```
double gValArray[100] = {
  0.000291608, 0.000347864, 0.000413481, 0.000489709, 0.000577906, 0.000679536, 0.000796168, 0.000929466, 0.00108118, 0.00125314,
  0.00144724, 0.00166538, 0.00190953, 0.00218159, 0.00248347, 0.00281695, 0.00318374, 0.00358535, 0.00402311, 0.0044981,
  0.0050111, 0.00556254, 0.00615248, 0.00678053, 0.00744584, 0.00814705, 0.00888226, 0.00964901, 0.0104443, 0.0112645,
  0.0121054, 0.0129624, 0.0138302, 0.0147031, 0.0155748, 0.016439, 0.0172887, 0.0181171, 0.0189169, 0.019681,
  0.0204025, 0.0210743, 0.0216901, 0.0222436, 0.0227293, 0.0231421, 0.0234778, 0.0237327, 0.0239042, 0.0239904,
  0.0239904, 0.0239042, 0.0237327, 0.0234778, 0.0231421, 0.0227293, 0.0222436, 0.0216901, 0.0210743, 0.0204025,
  0.019681, 0.0189169, 0.0181171, 0.0172887, 0.016439, 0.0155748, 0.0147031, 0.0138302, 0.0129624, 0.0121054,
  0.0112645, 0.0104443, 0.00964901, 0.00888226, 0.00814705, 0.00744584, 0.00678053, 0.00615248, 0.00556254, 0.0050111,
  0.0044981, 0.00402311, 0.00358535, 0.00318374, 0.00281695, 0.00248347, 0.00218159, 0.00190953, 0.00166538, 0.00144724,
  0.00125314, 0.00108118, 0.000929466, 0.000796168, 0.000679536, 0.000577906, 0.000489709, 0.000413481, 0.000347864, 0.000291608
};
```

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Relevant part of code

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Loop over gaussian bins

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Obtain bin center

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

ASU

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Calculate the square root of bin content

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal   = gCenterArray[i];
  GDouble gVal    = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Define ΔMass

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

ASU

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Define ΔMass

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

Standard deviation passed into program

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Define ΔMass

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

Displacement of center passed into program

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Convolution step

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

ASU

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Convolution step

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal  = gCenterArray[i];
  GDouble gVal   = sqrt(gValArray[i]);
  GDouble deltaM = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

Square root of gaussian

Convoluted Breit-Wigner

- Copied BrietWigner.cc to BWSmear.cc
- Modified code to include gaussian convolution

Convolution step

```
complex <GDouble> bwVal(0,0);
for( unsigned int i = 0; i < 100; i++ ){
  GDouble gcVal   = gCenterArray[i];
  GDouble gVal    = sqrt(gValArray[i]);
  GDouble deltaM  = gcVal*m_smear + m_centerShift;
  bwVal += gVal*bwCalc(mass+deltaM,mass1,mass2,m_mass0,m_width0,m_orbitL);
}

return(bwVal);
```

Standard Breit-Wigner calculation

Title

Title