

***Practically* Incorporating
Theoretical Innovations into
Experimental Amplitude Analyses**

Ryan Mitchell
Indiana University
ATHOS 2012 (Camogli)
June 21, 2012

The Issue: Previously...

The previous generation of amplitude analysis fitting tools:

- commonly assumed the “isobar model” with 2-body Breit-Wigner resonance decays
(for example, $\pi_1 \rightarrow \rho\pi$; $\rho \rightarrow \pi\pi$)
- were often easy to use
(just input the number of Breit-Wigner amplitudes and their parameters)
- but were also a sort of black box
(where in the code can you even find the Breit-Wigner form?)
- offered little flexibility to incorporate new amplitudes
(code had to be hacked to add new $\pi\pi$ S-waves, for example)
- were model-dependent with unquantifiable effects
(how much would answers change if we went beyond the isobar model?)

The Issue: Currently...

The current generation of amplitude analysis fitting tools:

- allow more flexibility when defining amplitudes
(in fact in some cases there are NO predefined amplitudes)
- force the user to do more work
(at least writing amplitudes and possibly also defining kinematics)
- but are therefore less of a black box
(amplitude definitions are made explicit)
- incorporate state of the art technology to increase fit speeds
(for example, Graphical Processor Units (GPU))
- allow systematic studies of model dependencies
(try the K-matrix, for example, and see how answers change versus the isobar model)

⇒ **there are no longer *technological or experimental* barriers to incorporating theoretical innovations into experimental analyses...**

The IU AmpTools Framework (*as an example*)

AmpTools is a set of C++ classes that can be used for amplitude analyses.

The user supplies classes that are derived from AmpTools classes:

DataReader: read data (any format, any experiment) and pack it into a Kinematics object

DataWriter: take a Kinematics object and write data

Amplitude: take kinematics and output a complex number (user supplies as many of these as needed)

PlotGenerator: make plots from fit results

Then the user writes applications (for fitting, MC generation, plotting, etc.) that use the above classes.

source: <http://sourceforge.net/projects/amptools/> (documentation in progress)

Writing Amplitudes in AmpTools (I)

A user-defined **BreitWigner** class, deriving from the **AmpTools Amplitude** class.

Class definition:

```
#ifdef GPU_ACCELERATION
void GPUBreitWigner_exec( dim3 dimGrid, dim3 dimBlock, GPU_AMP_PROTO,
                          GDouble mass0, GDouble width0,
                          int daughter1, int daughter2 );
#endif // GPU_ACCELERATION

class BreitWigner : public Amplitude{
public:
    BreitWigner() : Amplitude() { setDefaultStatus( true ); }
    BreitWigner( const AmpParameter& mass0,
                 const AmpParameter& width0,
                 int daughter1, int daughter2 );
    BreitWigner* newAmplitude( const vector< string >& args ) const;
    complex< GDouble > calcAmplitude( GDouble** pKin ) const;

#ifdef GPU_ACCELERATION
    void launchGPUKernel( dim3 dimGrid, dim3 dimBlock, GPU_AMP_PROTO ) const;
    bool isGPUEnabled() const { return true; }
#endif // GPU_ACCELERATION
};
```

for GPU
calculation

initialization

calculation

for GPU
calculation

Writing Amplitudes in AmpTools (II)

A user-defined **BreitWigner** class, deriving from the AmpTools **Amplitude** class.

Initialization methods:

```
BreitWigner*
BreitWigner::newAmplitude( const vector< string >& args ) const {

    AmpParameter mass(args[0]);
    AmpParameter width(args[1]);
    int daughter1(atoi(args[2].c_str()));
    int daughter2(atoi(args[3].c_str()));

    return new BreitWigner( mass, width, daughter1, daughter2 );
}

BreitWigner::BreitWigner( const AmpParameter& mass0,
                          const AmpParameter& width0,
                          int daughter1, int daughter2) :
    Amplitude(),
    m_mass0( mass0 ),
    m_width0( width0 ),
    m_daughter1(daughter1),
    m_daughter2(daughter2) {

    registerParameter( m_mass0 );
    registerParameter( m_width0 );
}
}
```

← initialized with mass,
width, and daughters

← mass and width are
variable parameters

Writing Amplitudes in AmpTools (III)

A user-defined **BreitWigner** class, deriving from the AmpTools **Amplitude** class.

Calculation method (CPU):

```
complex< GDouble >
BreitWigner::calcAmplitude( GDouble** pKin ) const {

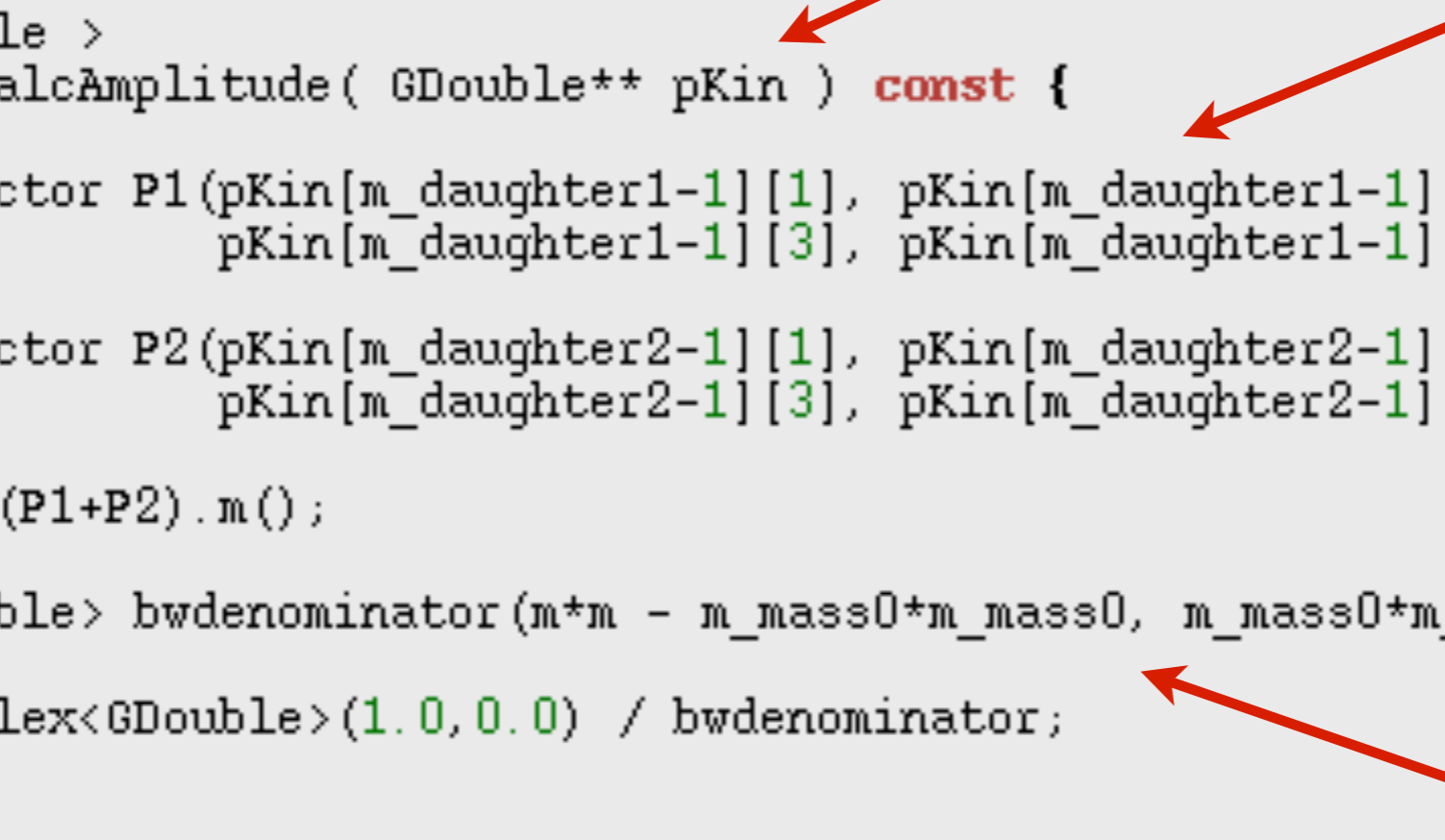
    HepLorentzVector P1(pKin[m_daughter1-1][1], pKin[m_daughter1-1][2],
                       pKin[m_daughter1-1][3], pKin[m_daughter1-1][0]);

    HepLorentzVector P2(pKin[m_daughter2-1][1], pKin[m_daughter2-1][2],
                       pKin[m_daughter2-1][3], pKin[m_daughter2-1][0]);

    GDouble m = (P1+P2).m();

    complex<GDouble> bwdenominator(m*m - m_mass0*m_mass0, m_mass0*m_width0);

    return complex<GDouble>(1.0,0.0) / bwdenominator;
}
```



(The GPU calculation, if enabled, is done in a separate function.)

Using Amplitudes with the AmpTools Framework (I)

Use “configuration files” to set up amplitudes:

```
reaction dalitz p1 p2 p3
sum dalitz s1
amplitude dalitz::s1::R12 BreitWigner 1.000 0.200 1 2
amplitude dalitz::s1::R13 BreitWigner 1.500 0.150 1 3
initialize dalitz::s1::R12 cartesian 1.0 0.0
initialize dalitz::s1::R13 cartesian 1.0 0.0 real
```

define a reaction

define a coherent sum

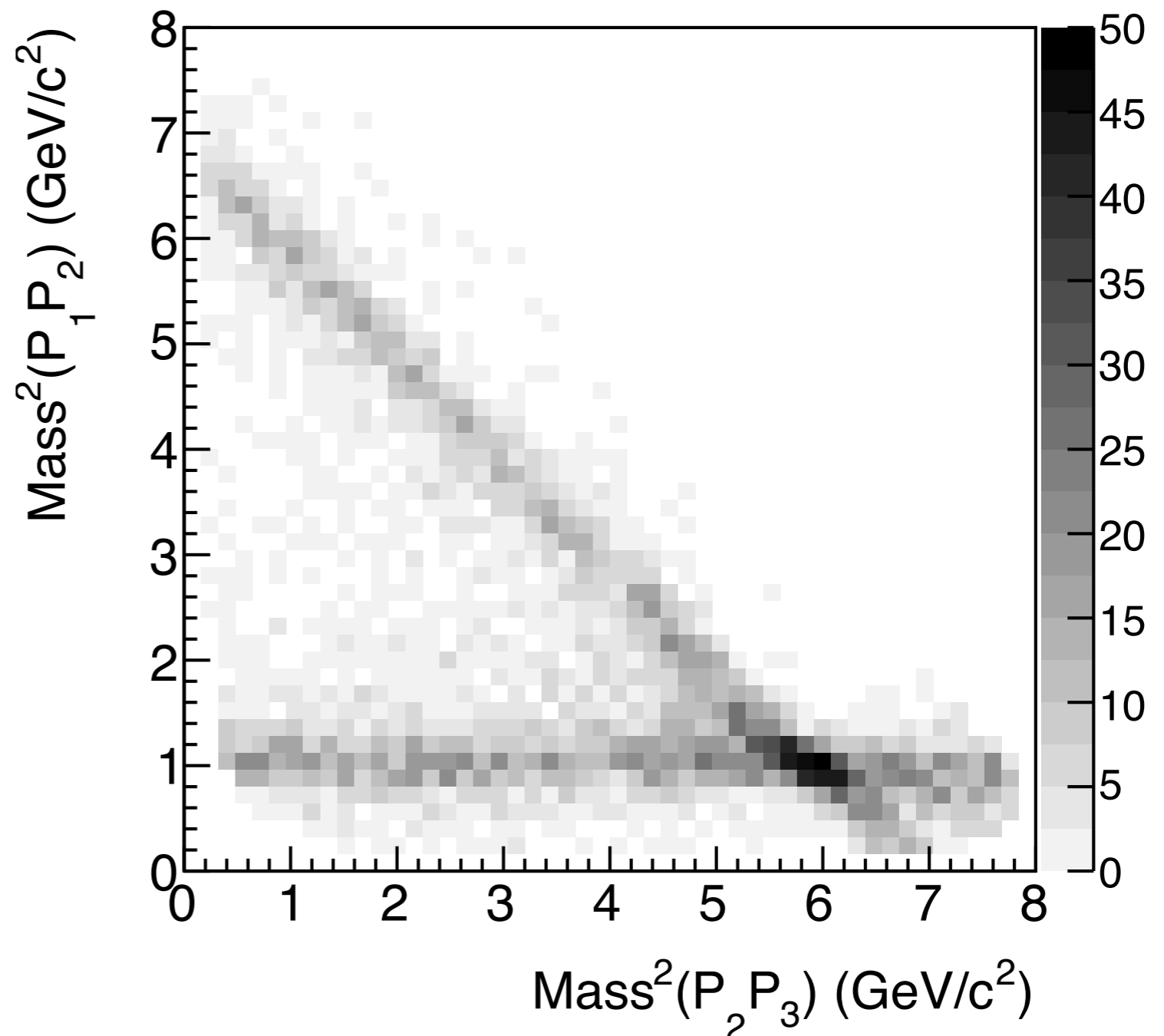
define two
amplitudes

initialize
amplitude
coefficients

Using Amplitudes with the AmpTools Framework (II)

Write a simple application to generate MC based on defined amplitudes
(using accept/reject on phase-space generated events):

*A fictitious decay of a
3.0 GeV/c² particle to
three 200 MeV/c² particles
according to the defined
amplitudes...*



Example Models of Implementation

(1) The Adam (Szczepaniak) Model:

(a theorist writing amplitudes in FORTRAN, then implemented in AmpTools by someone else (me))

(2) The Peng (Guo) Model:

(a theorist writing amplitudes in C++ and using AmpTools independently)

(3) The Mihajlo (Kornicer) Model:

(an experimentalist working closely with theorists to do an experimental analysis with AmpTools)

(1) The Adam Model: from FORTRAN...

excerpt of FORTRAN
code for Veneziano
 $\psi \rightarrow 3\pi$ amplitudes:

```
double complex function venez(n, m, s, t)
implicit double precision (a-h, o-z)
double complex alpha, cdgamma
venez =
$ cdgamma(n-alpha(s)) *
$ cdgamma(n-alpha(t))
$ /cdgamma(n+m-alpha(s)-alpha(t))
return
end

double complex function alpha(s)
implicit double precision (a-h, o-z)
common /mass/ xmp, xmX
if (s.gt.4.*xmp**2) then
xim = 0.280*dsqrt(s-4.*xmp**2)
else
xim = 0.d0
endif
alpha = 0.483 + 0.885*s + xim*(0.d0, 1.d0)
return
end

function cdgamma(x)
implicit real*8 (a - h, o - z)
complex*16 cdgamma, x
parameter (
& pi = 3.14159265358979324d+00,
& pv = 7.31790632447016203d+00,
& pu = 3.48064577727581257d+00,
& pr = 3.27673720261526849d-02,
```

(1) The Adam Model: ... to C++

after conversion to AmpTools framework:

input kinematics

```

complex< GDouble >
Veneziano::calcAmplitude( GDouble** pKin ) const {

    HepLorentzVector P1(pKin[0][1], pKin[0][2], pKin[0][3], pKin[0][0]);
    HepLorentzVector P2(pKin[1][1], pKin[1][2], pKin[1][3], pKin[1][0]);
    HepLorentzVector P3(pKin[2][1], pKin[2][2], pKin[2][3], pKin[2][0]);

    double s12 = (P1+P2).m2();
    double s13 = (P1+P3).m2();
    double s23 = (P2+P3).m2();
    double ph = sqrt(s13*s23*s12-(m_xmx*m_xmx-m_xmp*m_xmp)*(m_xmx*m_xmx-m_xmp*m_xmp)*m_xmp*m_xmp);

    return ((cdgamma(m_n-alpha(s13))*cdgamma(m_n-alpha(s23)))/cdgamma(m_n+m_m-alpha(s13)-alpha(s23))+
            (cdgamma(m_n-alpha(s13))*cdgamma(m_n-alpha(s12)))/cdgamma(m_n+m_m-alpha(s13)-alpha(s12))+
            (cdgamma(m_n-alpha(s23))*cdgamma(m_n-alpha(s12)))/cdgamma(m_n+m_m-alpha(s23)-alpha(s12)))
        * complex<GDouble>(ph, 0.0);
}

complex< GDouble >
Veneziano::alpha(double s) const{
    double xim = 0.280*sqrt(s-4.*m_xmp*m_xmp);
    return complex<GDouble> ((0.483 + 0.885*s),xim);
}

complex< GDouble >
Veneziano::cdgamma(complex< GDouble > x) const{

    double pi = 3.14159265358979324e+00;
    double pv = 7.31790632447016203e+00;
    double pu = 3.48064577727581257e+00;
    double pr = 3.27673720261526849e-02;
    double p = 3.27673720261526849e-02;

}
    
```

pack kinematics
into 4-vectors

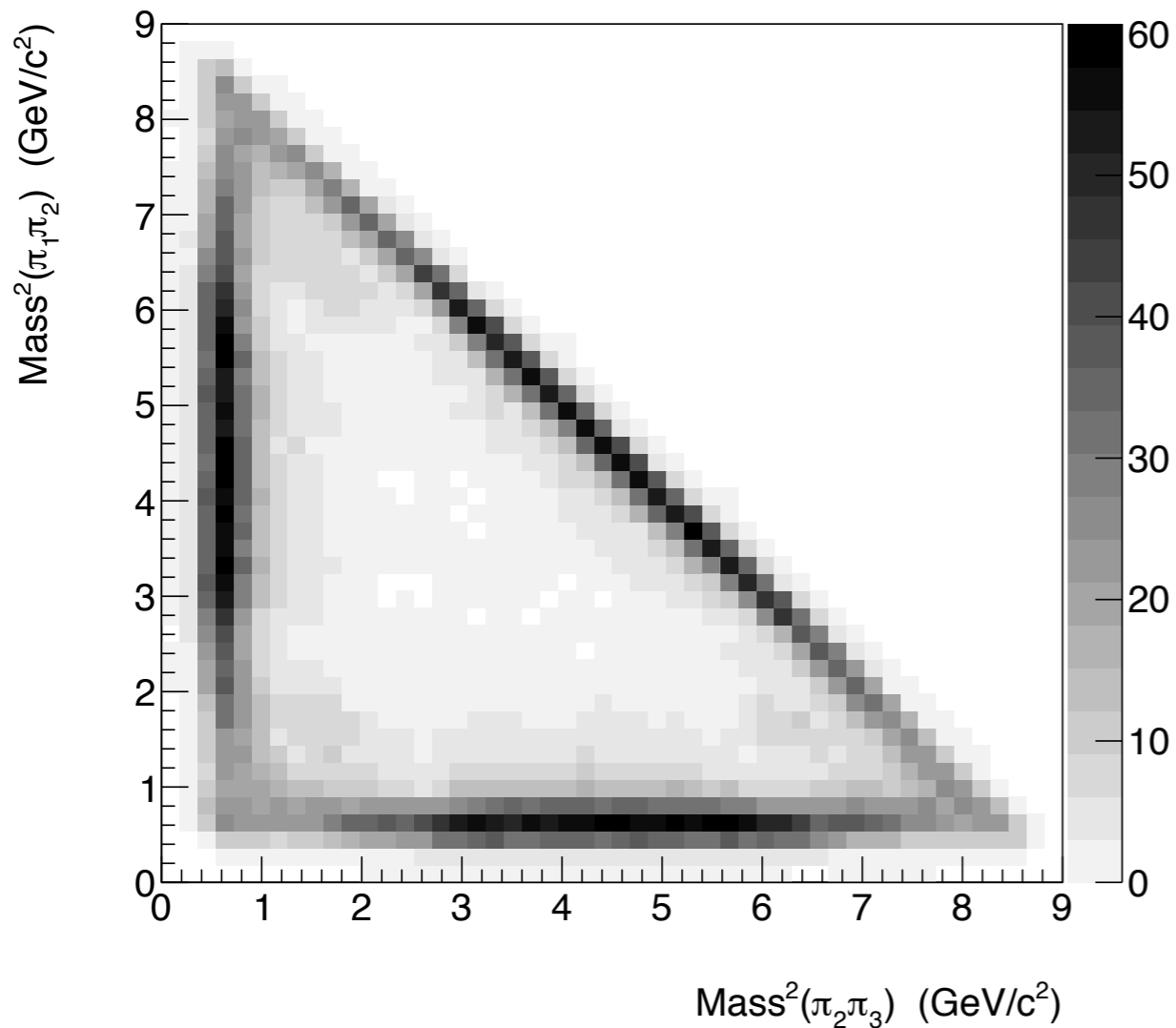
do the calculation

helper functions

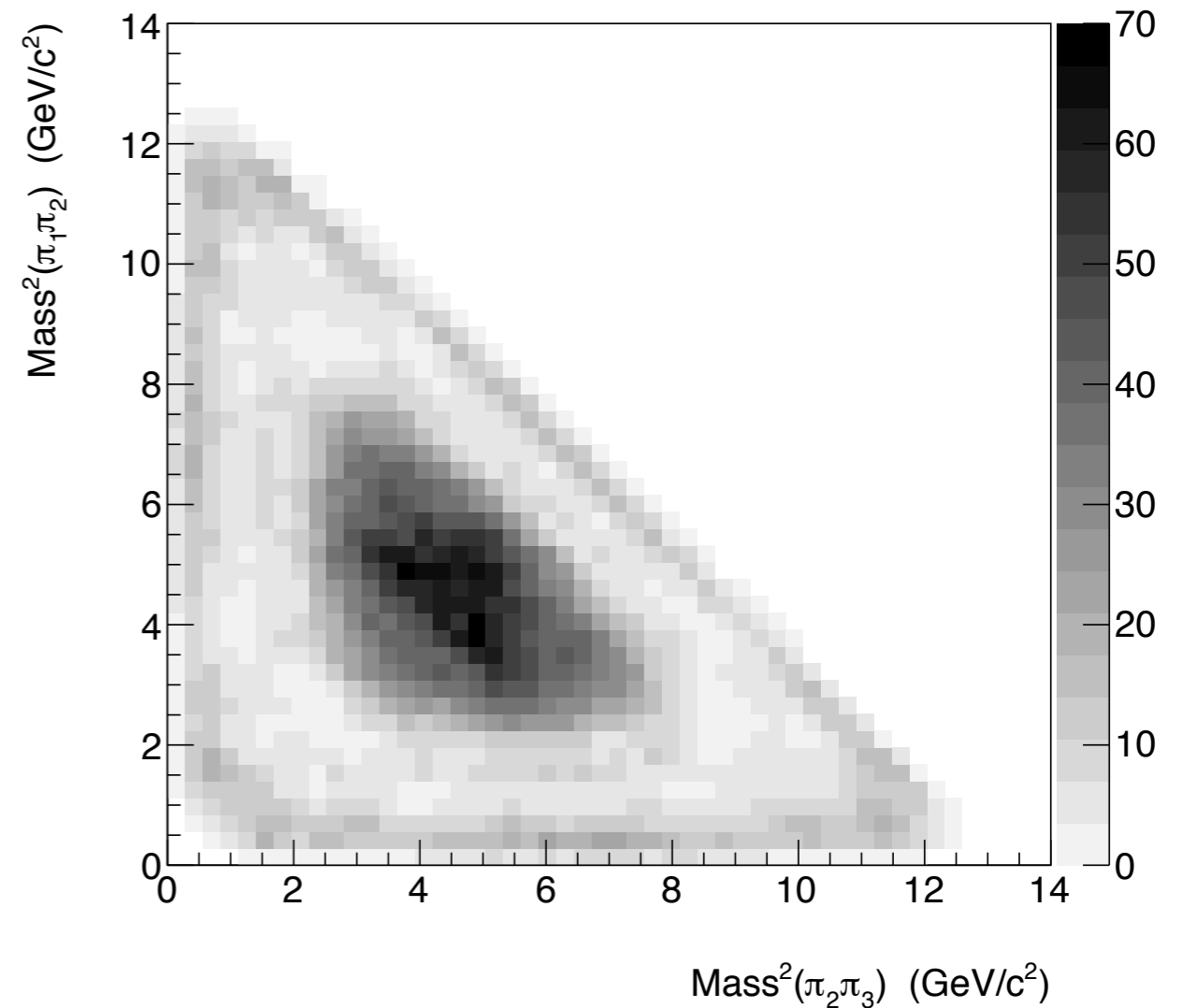
(1) The Adam Model: plots from AmpTools

generate distributions using AmpTools and fit parameters provided by Adam:

$J/\psi \rightarrow 3\pi$ using Veneziano Amplitudes



$\psi(2S) \rightarrow 3\pi$ using Veneziano Amplitudes



(2) The Peng (Guo) Model: theorist coding in C++

PHYSICAL REVIEW D **82**, 094002 (2010)

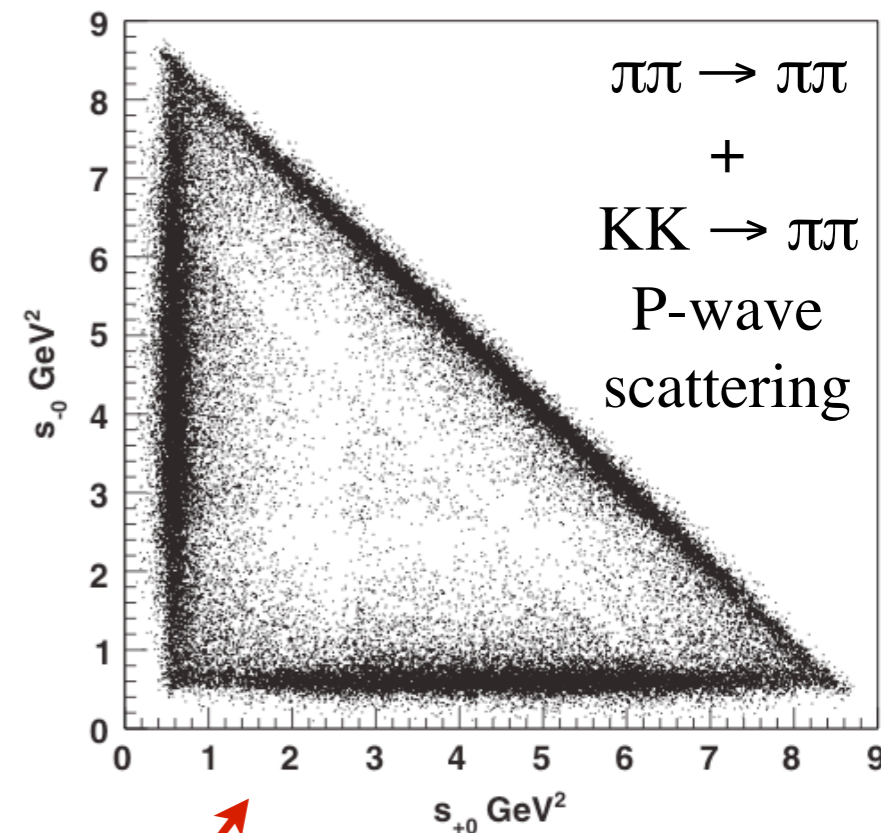
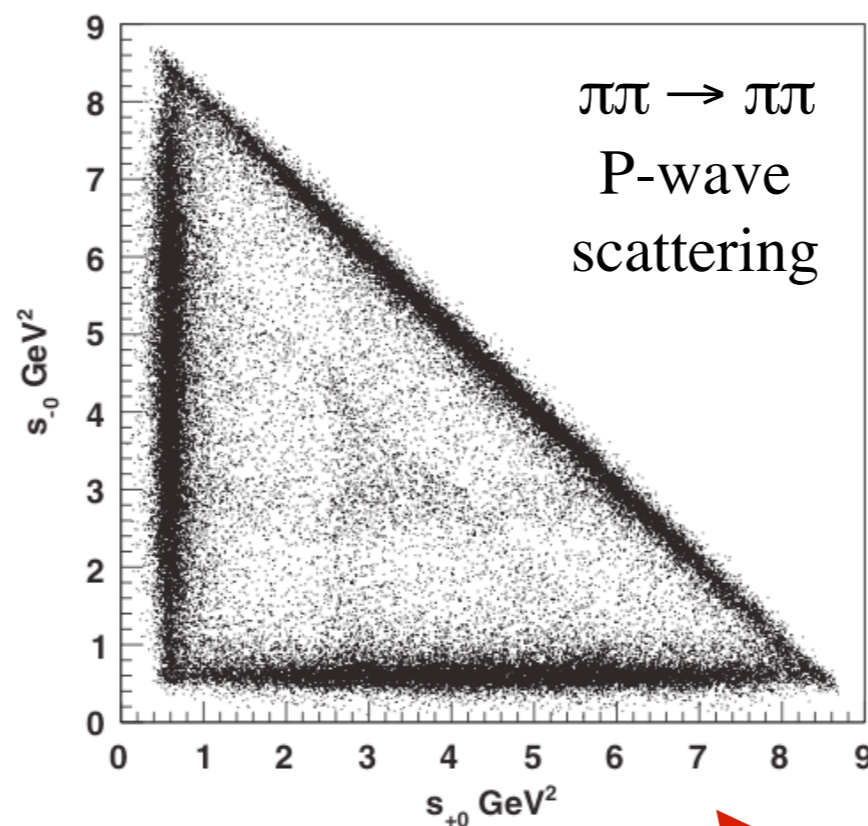
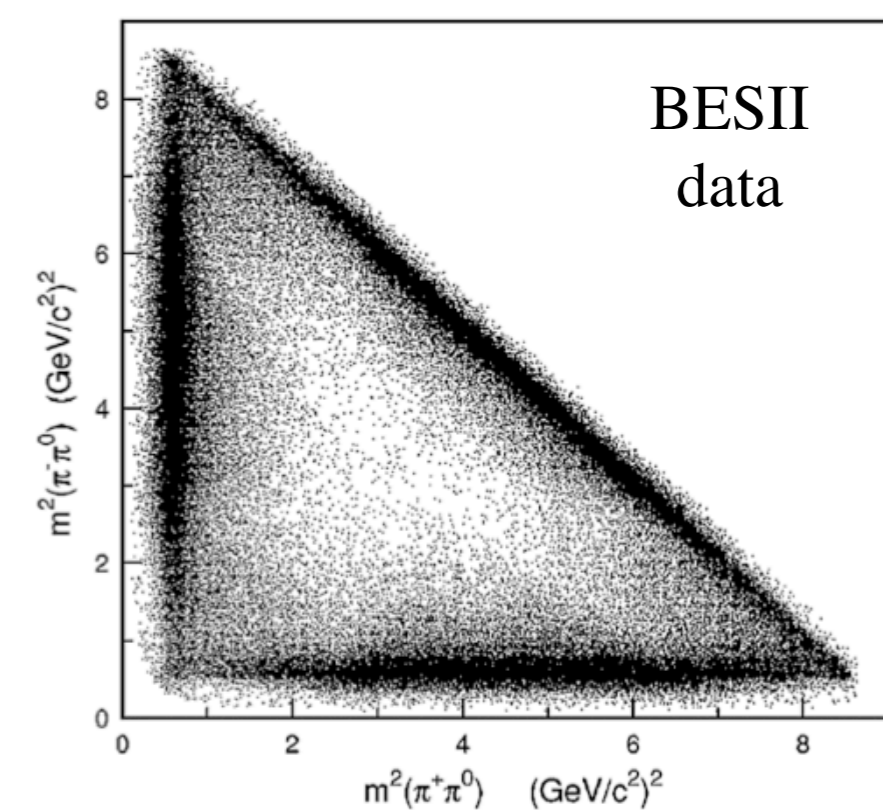
Role of P -wave inelasticity in $J/\psi \rightarrow \pi^+ \pi^- \pi^0$

Peng Guo,¹ Ryan Mitchell,¹ and Adam P. Szczepaniak^{1,2}

¹Physics Department, Indiana University, Bloomington, Indiana 47405, USA

²Center for the Exploration of Energy and Matter, Indiana University, Bloomington, Indiana 47408, USA

(Received 17 July 2010; published 5 November 2010)



amplitudes developed and coded by Peng Guo

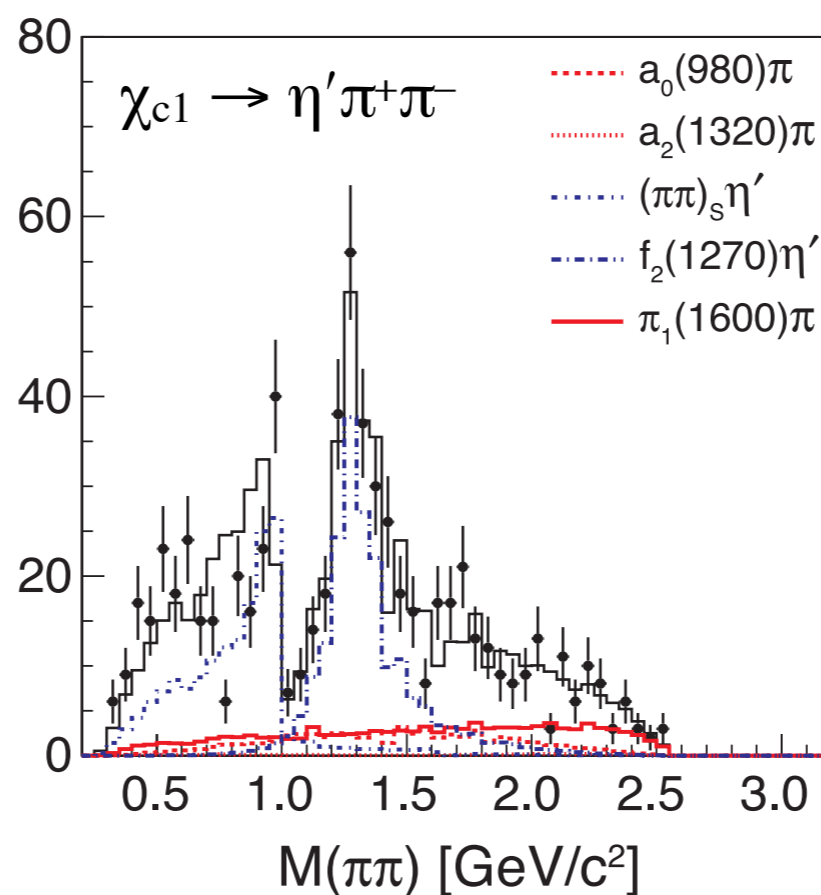
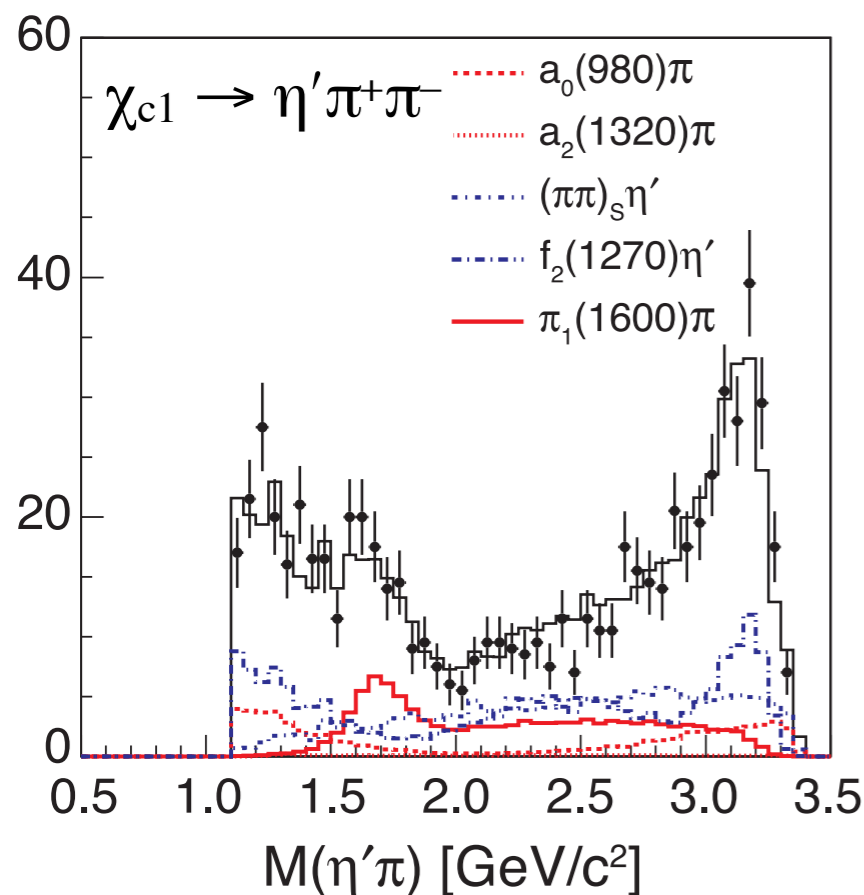
(3) The Mihajlo (Kornicer) Model: partial theory help

PHYSICAL REVIEW D **84**, 112009 (2011)

Amplitude analyses of the decays $\chi_{c1} \rightarrow \eta \pi^+ \pi^-$ and $\chi_{c1} \rightarrow \eta' \pi^+ \pi^-$

G. S. Adams,¹ J. Napolitano,¹ K. M. Ecklund,² J. Insler,³ H. Muramatsu,³ C. S. Park,³ L. J. Pearson,³ E. H. Thorndike,³ S. Ricciardi,⁴ C. Thomas,^{4,5} M. Artuso,⁶ S. Blusk,⁶ R. Mountain,⁶ T. Skwarnicki,⁶ S. Stone,⁶ L. M. Zhang,⁶ G. Bonvicini,⁷ D. Cinabro,⁷ A. Lincoln,⁷ M. J. Smith,⁷ P. Zhou,⁷ J. Zhu,⁷ P. Naik,⁸ J. Rademacker,⁸ D. M. Asner,^{9,*} K. W. Edwards,⁹ K. Randrianarivony,⁹ G. Tatishvili,^{9,*} R. A. Briere,¹⁰ H. Vogel,¹⁰ P. U. E. Onyisi,¹¹ J. L. Rosner,¹¹ J. P. Alexander,¹² D. G. Cassel,¹² S. Das,¹² R. Ehrlich,¹² L. Gibbons,¹² S. W. Gray,¹² D. L. Hartill,¹² B. K. Heltsley,¹² D. L. Kreinick,¹² V. E. Kuznetsov,¹² J. R. Patterson,¹² D. Peterson,¹² D. Riley,¹² A. Ryd,¹² A. J. Sadoff,¹² X. Shi,¹² W. M. Sun,¹² J. Yelton,¹³ P. Rubin,¹⁴ N. Lowrey,¹⁵ S. Mehrabyan,¹⁵ M. Selen,¹⁵ J. Wiss,¹⁵ J. Libby,¹⁶ M. Kornicer,¹⁷ R. E. Mitchell,¹⁷ M. R. Shepherd,¹⁷ A. Szczepaniak,¹⁷ D. Besson,¹⁸ T. K. Pedlar,¹⁹ D. Cronin-Hennessy,²⁰ J. Hietala,²⁰ S. Dobbs,²¹ Z. Metreveli,²¹ K. K. Seth,²¹ A. Tomaradze,²¹ T. Xiao,²¹ L. Martin,⁵ A. Powell,⁵ G. Wilkinson,⁵ J. Y. Ge,²² D. H. Miller,²² I. P. J. Shipsey,²² and B. Xin²²

(CLEO Collaboration)



Two-body amplitudes:

- (1) $\pi\pi \rightarrow \pi\pi$ and $KK \rightarrow \pi\pi$
S-wave scattering (*from theory*)
- (2) 3-channel Flatté formalism for $a_0(980) \rightarrow \eta'\pi$
- (3) Breit-Wigner forms for other resonances (*including first evidence for an exotic state in charmonium decays*)

Summary

We've reached an important milestone:

⇒ **there are no longer *technological* or *experimental* barriers to incorporating theoretical innovations into experimental analyses**

(but of course there is always room for improvement)

⇒ **the more pressing issue (in my opinion) is the further development of *theoretical* amplitudes for the description of experimental data**