Tensor pomeron model calculations – an object-oriented implementation

Krzysztof Cieśla
Paweł Buglewicz

Work supported in part by Polish National Science Centre grant
UMO-2015/17/D/ST2/03530

Kraków, 5 – 8 September 2017
Challenges in Photon Induced Interactions

Tensor pomeron model

- Recent works by P. Lebiedowicz, C. Ewertz, A. Szczurek and O. Nachtmann
- Allows consistent treatment of different processes
- Predictions agree with recent measurements of elastic scattering at RHIC using polarised proton beams

But formulas can be complicated:

$$\mathcal{M}^{(I\!P I\!P \to f_0 \to \pi^+\pi^-)}_{\lambda_a\lambda_b\to\lambda_1\lambda_2\pi^+\pi^-} = (-i)\bar{u}(p_1,\lambda_1) i\Gamma^{(I\!P pp)}_{\mu_1\nu_1}(p_1,p_a) u(p_a,\lambda_a)\times$$

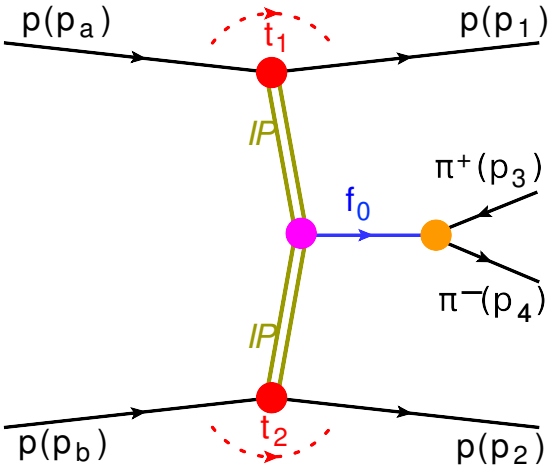
$$i\Delta^{(I\!P)\mu_1\nu_1,\alpha_1\beta_1}(s_1,t_1)\times i\Gamma^{(I\!P I\!P f_0)}_{\alpha_1\beta_1,\alpha_2\beta_2}(q_1,q_2)\times i\Delta^{(f_0)}(p_{34})\times i\Gamma^{(f_0\pi\pi)}(p_{34})\times$$

$$i\Delta^{(I\!P)\alpha_2\beta_2,\mu_2\nu_2}(s_2,t_2)\times\bar{u}(p_2,\lambda_2) i\Gamma^{(I\!P pp)}_{\mu_2\nu_2}(p_2,p_b) u(p_b,\lambda_b)$$

Thus, implementation of complex processes is prone to errors.

Solution: a library simplifying calculation of amplidtues for different processes.

Example: Pion Pair Production via f_0 resonance



- $pp \to p(f_0 \to \pi^+\pi^-)p$,
- model based on: Phys. Rev. D **93** (2016) 054015,
- tensor Pomeron exchange,

$$\mathcal{M}^{(IPIP\to f_0\to\pi^+\pi^-)}_{\lambda_a\lambda_b\to\lambda_1\lambda_2\pi^+\pi^-} =$$ matrix element

$$(-i)\bar{u}(p_1,\lambda_1)i\Gamma^{(IPpp)}_{\mu_1\nu_1}(p_1,p_a)u(p_a,\lambda_a)\times$$ proton-Pomeron vertex

$$i\Delta^{(IP)\mu_1\nu_1,\alpha_1\beta_1}(s_1,t_1)\times$$ Pomeron propagator

$$i\Gamma^{(IPIPf_0)}_{\alpha_1\beta_1,\alpha_2\beta_2}(q_1,q_2)\times$$ Pomeron-f_0 vertex

$$i\Delta^{(f_0)}(p_{34})\times$$ f_0 propagator

$$i\Gamma^{(f_0\pi\pi)}(p_{34})\times$$ f_0-pion vertex

$$i\Delta^{(IP)\alpha_2\beta_2,\mu_2\nu_2}(s_2,t_2)\times$$ Pomeron propagator

$$\bar{u}(p_2,\lambda_2)i\Gamma^{(IPpp)}_{\mu_2\nu_2}(p_2,p_b)u(p_b,\lambda_b)$$ Pomeron-proton vertex

Solution

Each vertex and propagator represented as class derived from corensponding base class:

- OScalar – scalar (eg. Pion propagator)
- OVector – 4 el. vector (eg. Proton-Photon vertex)
- OMatrix – 4 by 4 matrix (eg. Proton-Pomeron vertex)
- OTensor4 – rank 4 tensor (eg. Pomeron Propagator)

Overload operators (*, +, –), eg:

- OMatrix · OTensor4: $T_{ij} \cdot T^{ijkl} = T^{kl}$
- OTensor4 · OMatrix: $T^{ijkl} \cdot T_{kl} = T^{ij}$

Used tools: C++, ROOT (SVector, SMatrix, TLorentzVector classes).

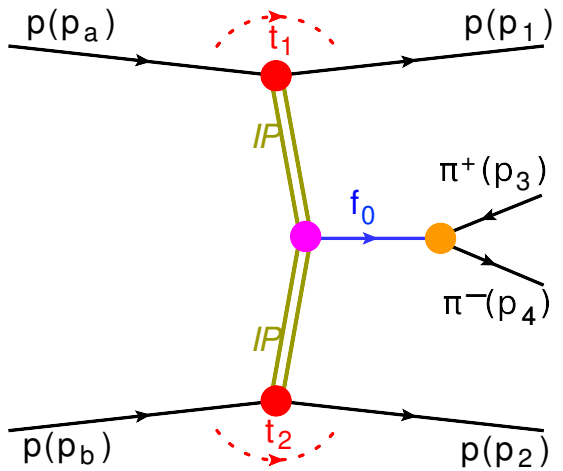
Example of internal implementation

$$i\Delta^{(\mathbb{P})}_{\mu\nu,\kappa\lambda}(s,t) = \frac{1}{4s}\left(g_{\mu\kappa}g_{\nu\lambda} + g_{\mu\lambda}g_{\nu\kappa} - \frac{1}{2}g_{\mu\nu}g_{\kappa\lambda}\right)(-is\alpha'_{\mathbb{P}})^{\alpha_{\mathbb{P}}(t)-1}$$

```
1  class PropPom : public OTensor4 {
2  [...]
3
4      for (int i = 0; i < 4; ++i)
5        for (int j = 0; j < 4; ++j)
6          for (int k = 0; k < 4; ++k)
7            for (int l = 0; l < 4; ++l)
8              result(i, j, k, l) = g(i, k) * g(j, l)
9                                  + g(i, l) * g(j, k)
10                                 - g(i, j) * g(k, l) / 2.0;
11
12 [..]
13 };
```

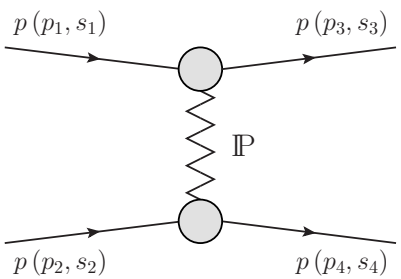
Amplitude calculation – user's perspective

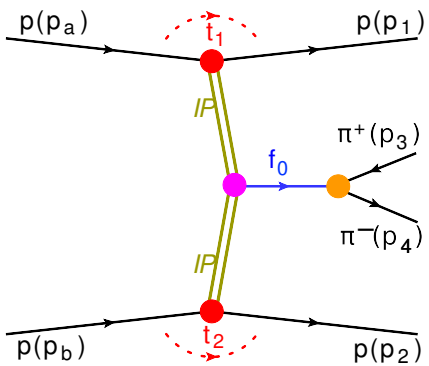
```
1  |    TLorentzVector pa, pb, p1, p2, p3, p4;
2  |    const std::complex<double> IU(0, 1);
3  |    [...]
4  |    double t1 = (pa-p1).Mag2();
5  |    TLorentzVector p34 = p3 + p4;
6  |    [...]
7  |    VtxPomProtProt V1(pa, p1, +1, -1);
8  |    VtxPomProtProt V2(pb, p2, +1, +1);
9  |    PropPom P1(s1, t1);
10 |    PropPom P2(s2, t2);
11 |
12 |    VtxPomPomf0 V3(q1, q2);
13 |    Propf0 P3(p34);
14 |    Vtxf0PiPi V4(p34);
15 |
16 |    std::complex<double> A;
17 |    A = -IU*V1*P1*V3*P3*V4*P2*V2;
```

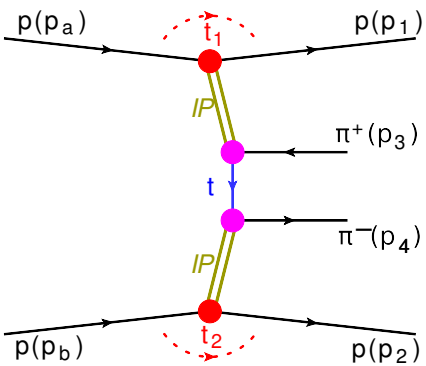


Status

- Tensor operations implemented
- Existing elements:
  - Pomeron–proton vertex
  - Pomeron propagator
  - Pomeron–pion vertex
  - Pion propagator
  - Pomeron–$f_0(980)$ vertex
  - $f_0(980)$ propagator
  - $f_0(980)$-pion vertex
- Existing processes:







- Next step: validate the results

Summary

- Goal: easy calculation of amplitudes in tensor pomeron model
- Object-oriented implementation in C++
- Using some classes from ROOT library
- Core of the library – tensor operations
- Library will provide classes used for calculation of amplitudes: vertices and propagators
- Implementation of new processes easy – multiplication of objects of appropriate classes
- Status: working, not yet validated
- To be integrated with GenEx generator

Work supported in part by Polish National Science Centre grant UMO-2015/17/D/ST2/03530.