DIPARTIMENTO DI FISICA ETTORE PANCINI
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
INFN NAPOLI

Loop Integrals Numerical Evaluation with LINE

Renato Maria Prisco

renatomaria.prisco@unina.it

Standard Model at LHC 2025, April 7-10, Durham, UK

in collaboration with:

Jonathan Ronca (unipd & INFN), Francesco Tramontano (unina & INFN)

arXiv:2501.01943

https://github.com/line-git/line

Motivations

Motivations

The computation of **loop integrals** is one of the main bottlenecks in **higher-order calculations**, which are crucial for high-energy physics. Possible approaches are:

- **fully analytical:** only possible for a limited number of cases
- **hybrid analytical-numerical:** difficult to generalize
- **fully numerical:**

  - `pySecDec`
  - `FIESTA`
  - `AMFlow`
  - `...`

  computation of single point

  require relatively large computational resources

  need nothing else

  - `DESS`
  - `DiffExp`
  - `SeaSyde`
  - `...`

  computation of series expansion

  require DEs w.r.t. kinematics

  require boundary conditions

  relatively much more efficient

  can be iterated to propagate across the phase-space

no program to do both in a single tool

Motivations

The computation of **loop integrals** is one of the main bottlenecks in **higher-order calculations**, which are crucial for high-energy physics. Possible approaches are:

- **fully analytical:** only possible for a limited number of cases
- **hybrid analytical-numerical:** difficult to generalize
- **fully numerical:**

solving DEs via **series expansion** can serve **both** purposes!

- `pySecDec`
- `FIESTA`
- `AMFlow`
- `...`

also a **DE solver** (DEs w.r.t. auxiliary mass, internally computed)

computation of single point

require relatively large computational resources

need nothing else

- `DESS`
- `DiffExp`
- `SeaSyde`
- `...`

computation of series expansion

require DEs w.r.t. kinematics

require boundary conditions

relatively much more efficient

can be iterated to propagate across the phase-space

no program to do both in a single tool

What is LINE?

LINE (**L**oop **I**ntegrals **N**umerical **E**valuation) is a tool to compute **loop integrals** via **Differential Equations** (DEs)

boundary computation and
phase-space **propagation**
in a **single tool**

What is LINE?

LINE (**L**oop **I**ntegrals **N**umerical **E**valuation) is a tool to compute **loop integrals** via **Differential Equations** (DEs)

well, loop integrals... **and more!**

boundary computation and
phase-space **propagation**
in a **single tool**

- **integrals with delta functions** → phase-space integrals
- **integrals with linear propagators** → subtraction counterterms, EFT
- **special functions** → Chen iterated integrals, pentagon functions
- **gravitational waveform**

computable solving DEs

What is LINE?

LINE (**L**oop **I**ntegrals **N**umerical **E**valuation) is a tool to compute **loop integrals** via **Differential Equations** (DEs)

- **AMFlow** → DEs w.r.t. auxiliary mass (BCs at infinity)
- **DiffExp** → DEs via series expansion
- **SeaSyde** → DEs + complex masses

great and robust **Mathematica packages**

multi-purpose high-level software → not tailored for this use case

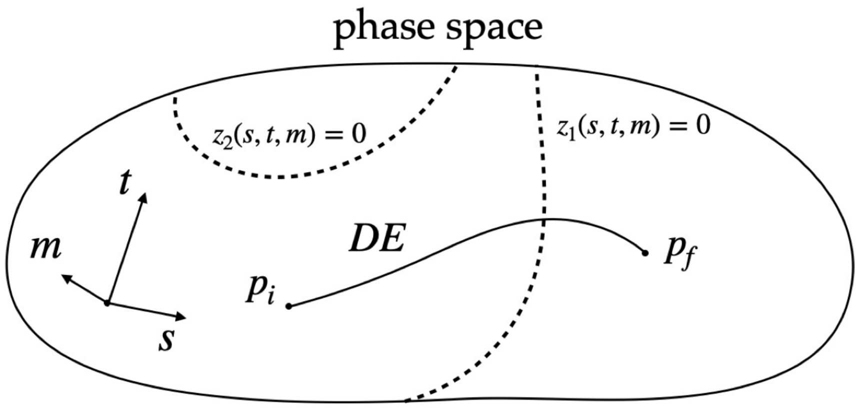
license issue → not ideal for massive cluster computations

LINE aims to improve on these aspects

faster low-level language (**C**) → optimal performance (only-what-we-need approach)

open source https://github.com/line-git/line → suitable for massive cluster computations

Loop Integrals via Differential Equations



- get **boundary conditions** at one point
- build DEs along the line connecting the **boundary** and the **target point**
- solve DEs via **series expansion**

$$\frac{d}{d\eta}\mathbf{I}(\eta) = \mathbb{A}(\eta)\mathbf{I}(\eta)$$

Master Integrals (MIs): set of independent Feynman integrals

solve for **numerical values** of ε and **interpolate (parallelization)**

ansatz around **regular** points

$$\mathbf{I}(\eta) = \sum_{k=0}^{\infty} \mathbf{I}_k \eta^k$$

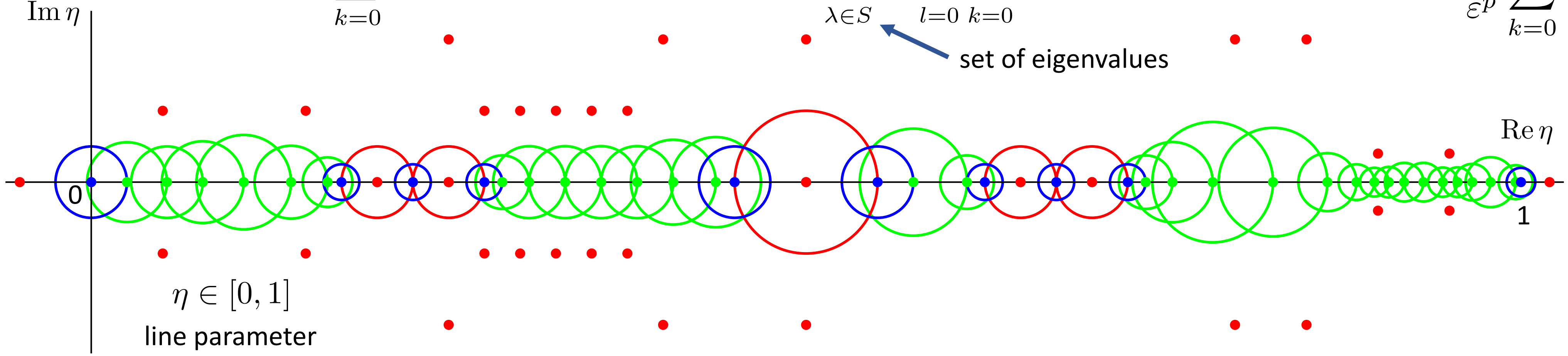
ansatz around **regular-singular** points

$$\mathbf{I}(\eta) = \sum_{\lambda\in S} \eta^\lambda \sum_{l=0}^{L_\lambda} \sum_{k=0}^{\infty} \mathbf{I}_{\lambda,l,k} \log^l(\eta)\eta^k$$

set of eigenvalues

$$\mathbf{I} = \frac{1}{\varepsilon^p}\sum_{k=0}^{\infty} \mathbf{I}_k \varepsilon^k \qquad d = 4 - 2\varepsilon$$

dimensional regularization

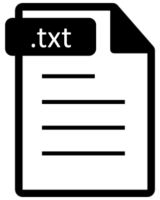


$\eta \in [0,1]$ line parameter

- **poles** of the DE matrix
- **regular** points
- **matching** points

Implementation

Mathematical Expressions in C



input file
DE matrix

Mathematical Expressions in C

```
2*(-4+d)^2*(10-7*d+d^2)*s^9*t^5*(3*s+4*t)*(3*(-3+d)*s+(-16+5*d)*t)-m2*s^7*t^4*(8-6*d+d^2)* (2*(-4150+3205*d-805*d^2+66*d^3)*s^4+(-
42520+32724*d-8219*d^2+675*d^3)*s^3*t+2*(-35500+27140*d-6743*d^2+532*d^3+3*d^4)*s^2*t^2+4*(-9914+7471*d-
1782*d^2+117*d^3+4*d^4)*s*t^3+8*(-50-149*d+148*d^2-43*d^3+4*d^4)*t^4)+262144*(-56+58*d-19*d^2+2*d^3)*m2^10*(s+t)*(4*(20-
9*d+d^2)*s^5+(535-272*d+33*d^2)*s^4*t+(1265-753*d+135*d^2-7*d^3)*s^3*t^2+(1797-1281*d+309*d^2-25*d^3)*s^2*t^3-2*(-638+525*d-
144*d^2+13*d^3)*s*t^4+4*(71-70*d+21*d^2-2*d^3)*t^5)+4*(-2+d)*m2^2*s^6*t^3*((7640-6653*d+2015*d^2-243*d^3+9*d^4)*s^5+4*(33430-
33161*d+12130*d^2-1942*d^3+115*d^4)*s^4*t+(444340-442838*d+161309*d^2-24828*d^3+1165*d^4+36*d^5)*s^3*t^2+2*(249308-
242356*d+83109*d^2-10606*d^3+2*d^4+67*d^5)*s^2*t^3+2*(69640-49054*d+2605*d^2+5514*d^3-1535*d^4+122*d^5)*s*t^4+4*(-
1048+7970*d-7887*d^2+3094*d^3-545*d^4+36*d^5)*t^5)+16*(-2+d)*m2^3*s^5*t^2*(6*(3100-3465*d+1444*d^2-
265*d^3+18*d^4)*s^6+(68620-79124*d+33975*d^2-6404*d^3+445*d^4)*s^5*t+(-29500+1205*d+15979*d^2-7473*d^3+1253*d^4-
72*d^5)*s^4*t^2-3*(69988-44332*d-1513*d^2+6687*d^3-1654*d^4+124*d^5)*s^3*t^3+(77798-240621*d+204195*d^2-
74961*d^3+12691*d^4-814*d^5)*s^2*t^4-8*(-35409+56641*d-35712*d^2+11089*d^3-1695*d^4+102*d^5)*s*t^5-4*(-8742+20263*d-
15499*d^2+5397*d^3-887*d^4+56*d^5)*t^6)-65536*m2^9*(2*(-7760+11692*d-6838*d^2+1952*d^3-273*d^4+15*d^5)*s^7+(-
133880+204796*d-121754*d^2+35335*d^3-5021*d^4+280*d^5)*s^6*t-2*(230470-363039*d+225204*d^2-69833*d^3+11179*d^4-
832*d^5+19*d^6)*s^5*t^2+(-881296+1460192*d-974010*d^2+335937*d^3-63289*d^4+6178*d^5-244*d^6)*s^4*t^3+(-1117108+1966668*d-
1416163*d^2+536687*d^3-113295*d^4+12661*d^5-586*d^6)*s^3*t^4-2*(425080-786430*d+597375*d^2-239205*d^3+53345*d^4-
6287*d^5+306*d^6)*s^2*t^5-4*(75164-145432*d+115125*d^2-47790*d^3+10987*d^4-1328*d^5+66*d^6)*s*t^6-8*(3976-8038*d+6585*d^2-
2802*d^3+655*d^4-80*d^5+4*d^6)*t^7)-64*m2^4*s^4*t*((-27440+43838*d-27405*d^2+8407*d^3-1267*d^4+75*d^5)*s^7+2*(-
137060+215942*d-132876*d^2+40069*d^3-5932*d^4+345*d^5)*s^6*t+(-830000+1325770*d-834559*d^2+262061*d^3-42123*d^4+3063*d^5-
60*d^6)*s^5*t^2+(-1369568+2316650*d-1582833*d^2+560776*d^3-108795*d^4+10962*d^5-448*d^6)*s^4*t^3-2*(1107084-
1971420*d+1439151*d^2-554278*d^3+119240*d^4-13619*d^5+646*d^6)*s^3*t^4+(-2689240+4874294*d-3633815*d^2+1433455*d^3-
316573*d^4+37173*d^5-1814*d^6)*s^2*t^5-2*(613448-1148204*d+884122*d^2-359659*d^3+81686*d^4-9831*d^5+490*d^6)*s*t^6-
4*(25760-49326*d+38467*d^2-15660*d^3+3517*d^4-414*d^5+20*d^6)*t^7)+16384*m2^8*(4*(-3340+5218*d-3180*d^2+949*d^3-
139*d^4+8*d^5)*s^8+(-168680+260686*d-156935*d^2+46219*d^3-6677*d^4+379*d^5)*s^7*t+(-776760+1215342*d-
744523*d^2+225335*d^3-34281*d^4+2243*d^5-28*d^6)*s^6*t^2+(-1750824+2832518*d-1824311*d^2+597425*d^3-104127*d^4+9011*d^5-
292*d^6)*s^5*t^3+(-2527592+4334208*d-3021960*d^2+1102421*d^3-222816*d^4+23735*d^5-1044*d^6)*s^4*t^4-2*(1257572-
2284780*d+1704803*d^2-671531*d^3+147648*d^4-17202*d^5+830*d^6)*s^3*t^5+(-1357816+2585562*d-2022013*d^2+832937*d^3-
190839*d^4+23071*d^5-1150*d^6)*s^2*t^6-2*(142736-285392*d+232778*d^2-99235*d^3+23358*d^4-2883*d^5+146*d^6)*s*t^7-4*(3976-
8038*d+6585*d^2-2802*d^3+655*d^4-80*d^5+4*d^6)*t^8)+256*m2^5*s^3*(2*(-3400+5400*d-3354*d^2+1022*d^3-153*d^4+9*d^5)*s^8+(-
119920+186864*d-113586*d^2+33817*d^3-4943*d^4+284*d^5)*s^7*t+(-644040+1006058*d-614765*d^2+185243*d^3-27921*d^4+1779*d^5-
18*d^6)*s^6*t^2+(-1660112+2669004*d-1703084*d^2+549991*d^3-93801*d^4+7826*d^5-236*d^6)*s^5*t^3-2*(1458844-
2455646*d+1670234*d^2-590065*d^3+114572*d^4-11626*d^5+483*d^6)*s^4*t^4+(-3834288+6658564*d-4715934*d^2+1753646*d^3-
362739*d^4+39719*d^5-1804*d^6)*s^3*t^5-2*(1286796-2271580*d+1638513*d^2-621235*d^3+131070*d^4-14633*d^5+677*d^6)*s^2*t^6-
4*(108724-179486*d+117200*d^2-38339*d^3+6476*d^4-507*d^5+12*d^6)*s*t^7+8*(-680-6274*d+11075*d^2-7130*d^3+2215*d^4-
336*d^5+20*d^6)*t^8)+4096*m2^7*s*(4*(-2180+3146*d-1742*d^2+465*d^3-60*d^4+3*d^5)*s^8+2*(-14200+20880*d-11748*d^2+3163*d^3-
407*d^4+20*d^5)*s^7*t+(203300-309420*d+181227*d^2-50405*d^3+6257*d^4-155*d^5-20*d^6)*s^6*t^2+(1010576-1573724*d+954516*d^2-
282685*d^3+40814*d^4-2257*d^5-8*d^6)*s^5*t^3+2*(986890-1627267*d+1075037*d^2-364353*d^3+66629*d^4-
6194*d^5+226*d^6)*s^4*t^4+(2712280-4827004*d+3520554*d^2-1353979*d^3+290645*d^4-33096*d^5+1564*d^6)*s^3*t^5+(2270132-
4284872*d+3330633*d^2-1368343*d^3+313753*d^4-38077*d^5+1910*d^6)*s^2*t^6+16*(48454-97161*d+79799*d^2-34386*d^3+8206*d^4-
1029*d^5+53*d^6)*s*t^7+4*(18044-38504*d+33167*d^2-14795*d^3+3617*d^4-461*d^5+24*d^6)*t^8)-1024*m2^6*s^2*(8*(-2230+3441*d-
2069*d^2+609*d^3-88*d^4+5*d^5)*s^8+(-166360+254642*d-151519*d^2+44024*d^3-6265*d^4+350*d^5)*s^7*t+(-596680+937046*d-
578117*d^2+177600*d^3-27977*d^4+2018*d^5-42*d^6)*s^6*t^2+(-1084336+1798006*d-1197961*d^2+411803*d^3-77217*d^4+7505*d^5-
296*d^6)*s^5*t^3-2*(661356-1160694*d+833066*d^2-314989*d^3+66520*d^4-7468*d^5+349*d^6)*s^4*t^4+(-775816+1362616*d-
978084*d^2+369317*d^3-77690*d^4+8657*d^5-400*d^6)*s^3*t^5+2*(310184-624798*d+520701*d^2-229750*d^3+56511*d^4-
7331*d^5+391*d^6)*s^2*t^6+2*(351912-727320*d+616792*d^2-274703*d^3+67760*d^4-8775*d^5+466*d^6)*s*t^7+4*(23880-
58276*d+55626*d^2-26933*d^3+7046*d^4-951*d^5+52*d^6)*t^8)
```

input file
DE matrix

Mathematical Expressions in C

```
2*(-4+d)^2*(10-7*d+d^2)*s^9*t^5*(3*s+4*t)*(3*(-3+d)*s+(-16+5*d)*t)-m2*s^7*t^4*(8-6*d+d^2)*(2[illegible]-4150+3205*d-805*d^2+66*d^3)*s^4+(-
42520+32724*d-8219*d^2+675*d^3)*s^3*t+2*(-35500+27140*d-6743*d^2+532*d^3[illegible]*d^4)*s^2*t^2[illegible]4*(-9914+7471*d-
1782*d^2+117*d^3+4*d^4)*s*t^3+8*(-50-149*d+148*d^2-43*d^3+4*d^4)*t^4)+2[illegible]44*(-56+58*d-1[illegible]d^2+2*d^3)*m2^10*(s+t)*(4*(20-
9*d+d^2)*s^5+(535-272*d+33*d^2)*s^4*t+(1265-753*d+135*d^2-7*d^3)*s^3*t^2[illegible]1[illegible]97-1281*d+30[illegible]*d^2-25*d^3)*s^2*t^3-2*(-638+525*d-
144*d^2+13*d^3)*s*t^4+4*(71-70*d+21*d^2-2*d^3)*t^5)+4*(-2+d)*m2^2*s^6*t^3*[illegible]7640-[illegible]53*d[illegible]15*d^2-243*d^3+9*d^4)*s^5+4*(33430-
33161*d+12130*d^2-1942*d^3+115*d^4)*s^4*t+(444340-442838*d+161309*d^2-24828[illegible]165[illegible]36*d^5)*s^3*t^2+2*(249308-
242356*d+83109*d^2-10606*d^3+2*d^4+67*d^5)*s^2*t^3+2*(69640-49054*d+2605*d^2+5514*d^3-1[illegible]d^4+122*d^5)*s*t^4+4*(-
1048+7970*d-7887*d^2+3094*d^3-545*d^4+36*d^5)*t^5)+16*(-2+d)*m2^3*s^5*t^2*(6*(3100-3465*d+1[illegible]d^2-
265*d^3+18*d^4)*s^6+(68620-79124*d+33975*d^2-6404*d^3+445*d^4)*s^5*t+(-29500+1205*d+15979*d^2[illegible]3*d^3+1253*d^4-
72*d^5)*s^4*t^2-3*(69988-44332*d-1513*d^2+6687*d^3-1654*d^4+124*d^5)*s^3*t^3+(77798-240621*d+204195*d^2-
74961*d^3+12691*d^4-814*d^5)*s^2*t^4-8*(-35409+56641*d-35712*d^2+11089*d^3-1695*d^4+102*d^5)*s*t^5-4*(-8742+20263*d-
15499*d^2+5397*d^3-887*d^4+56*d^5)*t^6)-65536*m2^9*(2*(-7760+11692*d-6838*d^2+1952*d^3-273*d^4+15*d^5)*s^7+(-
133880+204796*d-121754*d^2+35335*d^3-5021*d^4+280*d^5)*s^6*t-2*(230470-363039*d+225204*d^2-69833*d^3+11179*d^4-
832*d^5+19*d^6)*s^5*t^2+(-881296+1460192*d-974010*d^2+335937*d^3-63289*d^4+6178*d^5-244*d^6)*s^4*t^3+(-1117108+1966668*d-
1416163*d^2+536687*d^3-113295*d^4+12661*d^5-586*d^6)*s^3*t^4-2*(425080-786430*d+597375*d^2-239205*d^3+53345*d^4-
6287*d^5+306*d^6)*s^2*t^5-4*(75164-145432*d+115125*d^2-47790*d^3+10987*d^4-1328*d^5+66*d^6)*s*t^6-8*(3976-8038*d+6585*d^2-
2802*d^3+655*d^4-80*d^5+4*d^6)*t^7)-64*m2^4*s^4*t*((-27440+43838*d-27405*d^2+8407*d^3-1267*d^4+75*d^5)*s^7+2*(-
137060+215942*d-132876*d^2+40069*d^3-5932*d^4+345*d^5)*s^6*t+(-830000+1325770*d-834559*d^2+262061*d^3-42123*d^4+3063*d^5-
60*d^6)*s^5*t^2+(-1369568+2316650*d-1582833*d^2+560776*d^3-108795*d^4+10962*d^5-448*d^6)*s^4*t^3-2*(1107084-
1971420*d+1439151*d^2-554278*d^3+119240*d^4-13619*d^5+646*d^6)*s^3*t^4+(-2689240+4874294*d-3633815*d^2+1433455*d^3-
316573*d^4+37173*d^5-1814*d^6)*s^2*t^5-2*(613448-1148204*d+884122*d^2-359659*d^3+81686*d^4-9831*d^5+490*d^6)*s*t^6-
4*(25760-49326*d+38467*d^2-15660*d^3+3517*d^4-414*d^5+20*d^6)*t^7)+16384*m2^8*(4*(-3340+5218*d-3180*d^2+949*d^3-
139*d^4+8*d^5)*s^8+(-168680+260686*d-156935*d^2+46219*d^3-6677*d^4+379*d^5)*s^7*t+(-776760+1215342*d-
744523*d^2+225335*d^3-34281*d^4+2243*d^5-28*d^6)*s^6*t^2+(-1750824+2832518*d-1824311*d^2+597425*d^3-104127*d^4+9011*d^5-
292*d^6)*s^5*t^3+(-2527592+4334208*d-3021960*d^2+1102421*d^3-222816*d^4+23735*d^5-1044*d^6)*s^4*t^4-2*(1257572-
2284780*d+1704803*d^2-671531*d^3+147648*d^4-17202*d^5+830*d^6)*s^3*t^5+(-1357816+2585562*d-2022013*d^2+832937*d^3-
190839*d^4+23071*d^5-1150*d^6)*s^2*t^6-2*(142736-285392*d+232778*d^2-99235*d^3+23358*d^4-2883*d^5+146*d^6)*s*t^7-4*(3976-
8038*d+6585*d^2-2802*d^3+655*d^4-80*d^5+4*d^6)*t^8)+256*m2^5*s^3*(2*(-3400+5400*d-3354*d^2+1022*d^3-153*d^4+9*d^5)*s^8+(-
119920+186864*d-113586*d^2+33817*d^3-4943*d^4+284*d^5)*s^7*t+(-644040+1006058*d-614765*d^2+185243*d^3-27921*d^4+1779*d^5-
18*d^6)*s^6*t^2+(-1660112+2669004*d-1703084*d^2+549991*d^3-93801*d^4+7826*d^5-236*d^6)*s^5*t^3-2*(1458844-
2455646*d+1670234*d^2-590065*d^3+114572*d^4-11626*d^5+483*d^6)*s^4*t^4+(-3834288+6658564*d-4715934*d^2+1753646*d^3-
362739*d^4+39719*d^5-1804*d^6)*s^3*t^5-2*(1286796-2271580*d+1638513*d^2-621235*d^3+131070*d^4-14633*d^5+677*d^6)*s^2*t^6-
4*(108724-179486*d+117200*d^2-38339*d^3+6476*d^4-507*d^5+12*d^6)*s*t^7+8*(-680-6274*d+11075*d^2-7130*d^3+2215*d^4-
336*d^5+20*d^6)*t^8)+4096*m2^7*s*(4*(-2180+3146*d-1742*d^2+465*d^3-60*d^4+3*d^5)*s^8+2*(-14200+20880*d-11748*d^2+3163*d^3-
407*d^4+20*d^5)*s^7*t+(203300-309420*d+181227*d^2-50405*d^3+6257*d^4-155*d^5-20*d^6)*s^6*t^2+(1010576-1573724*d+954516*d^2-
282685*d^3+40814*d^4-2257*d^5-8*d^6)*s^5*t^3+2*(986890-1627267*d+1075037*d^2-364353*d^3+66629*d^4-
6194*d^5+226*d^6)*s^4*t^4+(2712280-4827004*d+3520554*d^2-1353979*d^3+290645*d^4-33096*d^5+1564*d^6)*s^3*t^5+(2270132-
4284872*d+3330633*d^2-1368343*d^3+313753*d^4-38077*d^5+1910*d^6)*s^2*t^6+16*(48454-97161*d+79799*d^2-34386*d^3+8206*d^4-
1029*d^5+53*d^6)*s*t^7+4*(18044-38504*d+33167*d^2-14795*d^3+3617*d^4-461*d^5+24*d^6)*t^8)-1024*m2^6*s^2*(8*(-2230+3441*d-
2069*d^2+609*d^3-88*d^4+5*d^5)*s^8+(-166360+254642*d-151519*d^2+44024*d^3-6265*d^4+350*d^5)*s^7*t+(-596680+937046*d-
578117*d^2+177600*d^3-27977*d^4+2018*d^5-42*d^6)*s^6*t^2+(-1084336+1798006*d-1197961*d^2+411803*d^3-77217*d^4+7505*d^5-
296*d^6)*s^5*t^3-2*(661356-1160694*d+833066*d^2-314989*d^3+66520*d^4-7468*d^5+349*d^6)*s^4*t^4+(-775816+1362616*d-
978084*d^2+369317*d^3-77690*d^4+8657*d^5-400*d^6)*s^3*t^5+2*(310184-624798*d+520701*d^2-229750*d^3+56511*d^4-
7331*d^5+391*d^6)*s^2*t^6+2*(351912-727320*d+616792*d^2-274703*d^3+67760*d^4-8775*d^5+466*d^6)*s*t^7+4*(23880-
58276*d+55626*d^2-26933*d^3+7046*d^4-951*d^5+52*d^6)*t^8)
```

```
m2*s^7*t^4*(8-6*d+d^2)
```

string representing a **mathematical expression**

input file
DE matrix

Mathematical Expressions in C

```
2*(-4+d)^2*(10-7*d+d^2)*s^9*t^5*(3*s+4*t)*(3*(-3+d)*s+(-16+5*d)*t)-m2*s^7*t^4*(8-6*d+d^2)*(2[illegible]4150+3205*d-805*d^2+66*d^3)*s^4+(-
42520+32724*d-8219*d^2+675*d^3)*s^3*t+2*(-35500+27140*d-6743*d^2+532*[illegible]d^4)*s^2*t^2+4*(-9914+7471*d-
1782*d^2+117*d^3+4*d^4)*s*t^3+8*(-50-149*d+148*d^2-43*d^3+4*d^4)*t^4)+2[illegible]44*(-56+58*d-1[illegible]d^2+2*d^3)*m2^10*(s+t)*(4*(20-
9*d+d^2)*s^5+(535-272*d+33*d^2)*s^4*t+(1265-753*d+135*d^2-7*d^3)*s^3*t^2+(1797-1281*d+30[illegible]*d^2-25*d^3)*s^2*t^3-2*(-638+525*d-
144*d^2+13*d^3)*s*t^4+4*(71-70*d+21*d^2-2*d^3)*t^5)+4*(-2+d)*m2^2*s^6*t^3*(7640-6053*d[illegible]15*d^2-243*d^3+9*d^4)*s^5+4*(33430-
33161*d+12130*d^2-1942*d^3+115*d^4)*s^4*t+(444340-442838*d+161309*d^2-24828[illegible]165[illegible]36*d^5)*s^3*t^2+2*(249308-
242356*d+83109*d^2-10606*d^3+2*d^4+67*d^5)*s^2*t^3+2*(69640-49054*d+2605*d^2+5514*d^3-1[illegible]d^4+122*d^5)*s*t^4+4*(-
1048+7970*d-7887*d^2+3094*d^3-545*d^4+36*d^5)*t^5)+16*(-2+d)*m2^3*s^5*t^2*(6*(3100-3465*d+1[illegible]d^2-
265*d^3+18*d^4)*s^6+(68620-79124*d+33975*d^2-6404*d^3+445*d^4)*s^5*t+(-29500+1205*d+15979*d^2[illegible]3*d^3+1253*d^4-
72*d^5)*s^4*t^2-3*(69988-44332*d-1513*d^2+6687*d^3-1654*d^4+124*d^5)*s^3*t^3+(77798-240621*d+204195*d^2-
74961*d^3+12691*d^4-814*d^5)*s^2*t^4-8*(-35409+56641*d-35712*d^2+11089*d^3-1695*d^4+102*d^5)*s*t^5-4*(-8742+20263*d-
15499*d^2+5397*d^3-887*d^4+56*d^5)*t^6)-65536*m2^9*(2*(-7760+11692*d-6838*d^2+1952*d^3-273*d^4+15*d^5)*s^7+(-
133880+204796*d-121754*d^2+35335*d^3-5021*d^4+280*d^5)*s^6*t-2*(230470-363039*d+225204*d^2-69833*d^3+11179*d^4-
832*d^5+19*d^6)*s^5*t^2+(-881296+1460192*d-974010*d^2+335937*d^3-63289*d^4+6178*d^5-244*d^6)*s^4*t^3+(-1117108+1966668*d-
1416163*d^2+536687*d^3-113295*d^4+12661*d^5-586*d^6)*s^3*t^4-2*(425080-786430*d+597375*d^2-239205*d^3+53345*d^4-
6287*d^5+306*d^6)*s^2*t^5-4*(75164-145432*d+115125*d^2-47790*d^3+10987*d^4-1328*d^5+66*d^6)*s*t^6-8*(3976-8038*d+6585*d^2-
2802*d^3+655*d^4-80*d^5+4*d^6)*t^7)-64*m2^4*s^4*t*((-27440+43838*d-27405*d^2+8407*d^3-1267*d^4+75*d^5)*s^7+2*(-
137060+215942*d-132876*d^2+40069*d^3-5932*d^4+345*d^5)*s^6*t+(-830000+1325770*d-834559*d^2+262061*d^3-42123*d^4+3063*d^5-
60*d^6)*s^5*t^2+(-1369568+2316650*d-1582833*d^2+560776*d^3-108795*d^4+10962*d^5-448*d^6)*s^4*t^3-2*(1107084-
1971420*d+1439151*d^2-554278*d^3+119240*d^4-13619*d^5+646*d^6)*s^3*t^4+(-2689240+4874294*d-3633815*d^2+1433455*d^3-
316573*d^4+37173*d^5-1814*d^6)*s^2*t^5-2*(613448-1148204*d+884122*d^2-359659*d^3+81686*d^4-9831*d^5+490*d^6)*s*t^6-
4*(25760-49326*d+38467*d^2-15660*d^3+3517*d^4-414*d^5+20*d^6)*t^7)+16384*m2^8*(4*(-3340+5218*d-3180*d^2+949*d^3-
139*d^4+8*d^5)*s^8+(-168680+260686*d-156935*d^2+46219*d^3-6677*d^4+379*d^5)*s^7*t+(-776760+1215342*d-
744523*d^2+225335*d^3-34281*d^4+2243*d^5-28*d^6)*s^6*t^2+(-1750824+2832518*d-1824311*d^2+597425*d^3-104127*d^4+9011*d^5-
292*d^6)*s^5*t^3+(-2527592+4334208*d-3021960*d^2+1102421*d^3-222816*d^4+23735*d^5-1044*d^6)*s^4*t^4-2*(1257572-
2284780*d+1704803*d^2-671531*d^3+147648*d^4-17202*d^5+830*d^6)*s^3*t^5+(-1357816+2585562*d-2022013*d^2+832937*d^3-
190839*d^4+23071*d^5-1150*d^6)*s^2*t^6-2*(142736-285392*d+232778*d^2-99235*d^3+23358*d^4-2883*d^5+146*d^6)*s*t^7-4*(3976-
8038*d+6585*d^2-2802*d^3+655*d^4-80*d^5+4*d^6)*t^8)+256*m2^5*s^3*(2*(-3400+5400*d-3354*d^2+1022*d^3-153*d^4+9*d^5)*s^8+(-
119920+186864*d-113586*d^2+33817*d^3-4943*d^4+284*d^5)*s^7*t+(-644040+1006058*d-614765*d^2+185243*d^3-27921*d^4+1779*d^5-
18*d^6)*s^6*t^2+(-1660112+2669004*d-1703084*d^2+549991*d^3-93801*d^4+7826*d^5-236*d^6)*s^5*t^3-2*(1458844-
2455646*d+1670234*d^2-590065*d^3+114572*d^4-11626*d^5+483*d^6)*s^4*t^4+(-3834288+6658564*d-4715934*d^2+1753646*d^3-
362739*d^4+39719*d^5-1804*d^6)*s^3*t^5-2*(1286796-2271580*d+1638513*d^2-621235*d^3+131070*d^4-14633*d^5+677*d^6)*s^2*t^6-
4*(108724-179486*d+117200*d^2-38339*d^3+6476*d^4-507*d^5+12*d^6)*s*t^7+8*(-680-6274*d+11075*d^2-7130*d^3+2215*d^4-
336*d^5+20*d^6)*t^8)+4096*m2^7*s*(4*(-2180+3146*d-1742*d^2+465*d^3-60*d^4+3*d^5)*s^8+2*(-14200+20880*d-11748*d^2+3163*d^3-
407*d^4+20*d^5)*s^7*t+(203300-309420*d+181227*d^2-50405*d^3+6257*d^4-155*d^5-20*d^6)*s^6*t^2+(1010576-1573724*d+954516*d^2-
282685*d^3+40814*d^4-2257*d^5-8*d^6)*s^5*t^3+2*(986890-1627267*d+1075037*d^2-364353*d^3+66629*d^4-
6194*d^5+226*d^6)*s^4*t^4+(2712280-4827004*d+3520554*d^2-1353979*d^3+290645*d^4-33096*d^5+1564*d^6)*s^3*t^5+(2270132-
4284872*d+3330633*d^2-1368343*d^3+313753*d^4-38077*d^5+1910*d^6)*s^2*t^6+16*(48454-97161*d+79799*d^2-34386*d^3+8206*d^4-
1029*d^5+53*d^6)*s*t^7+4*(18044-38504*d+33167*d^2-14795*d^3+3617*d^4-461*d^5+24*d^6)*t^8)-1024*m2^6*s^2*(8*(-2230+3441*d-
2069*d^2+609*d^3-88*d^4+5*d^5)*s^8+(-166360+254642*d-151519*d^2+44024*d^3-6265*d^4+350*d^5)*s^7*t+(-596680+937046*d-
578117*d^2+177600*d^3-27977*d^4+2018*d^5-42*d^6)*s^6*t^2+(-1084336+1798006*d-1197961*d^2+411803*d^3-77217*d^4+7505*d^5-
296*d^6)*s^5*t^3-2*(661356-1160694*d+833066*d^2-314989*d^3+66520*d^4-7468*d^5+349*d^6)*s^4*t^4+(-775816+1362616*d-
978084*d^2+369317*d^3-77690*d^4+8657*d^5-400*d^6)*s^3*t^5+2*(310184-624798*d+520701*d^2-229750*d^3+56511*d^4-
7331*d^5+391*d^6)*s^2*t^6+2*(351912-727320*d+616792*d^2-274703*d^3+67760*d^4-8775*d^5+466*d^6)*s*t^7+4*(23880-
58276*d+55626*d^2-26933*d^3+7046*d^4-951*d^5+52*d^6)*t^8)
```

m2*s^7*t^4*(8-6*d+d^2)

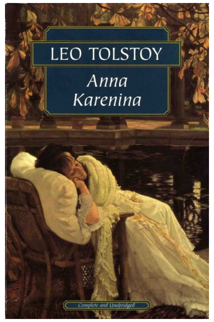
string representing a **mathematical expression**

input file
DE matrix

for a single matrix element:

2 MB ~

(~ 1000 pages)

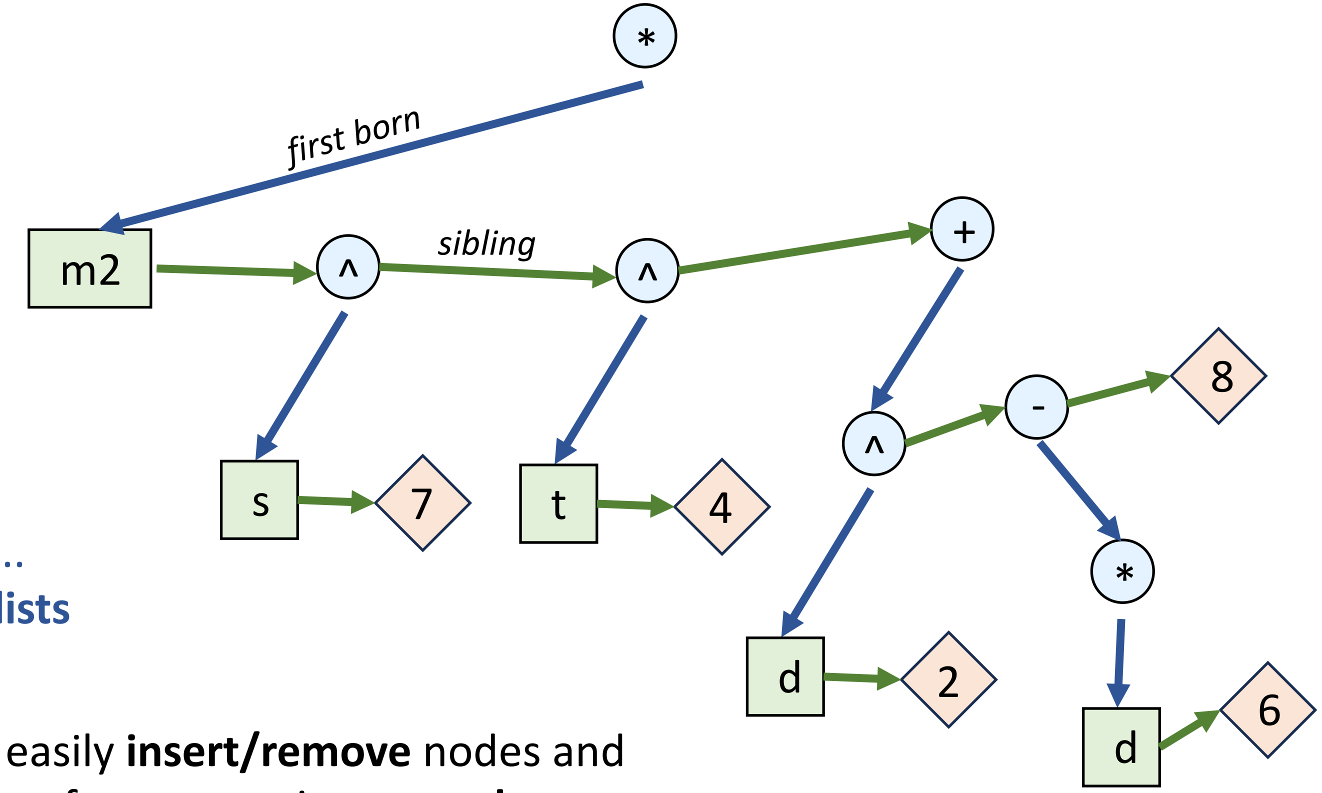


Mathematical Expressions in C

The DE matrix elements in the input files are **parsed** and converted to **symbolic trees**

m2*s^7*t^4*(8-6*d+d^2)

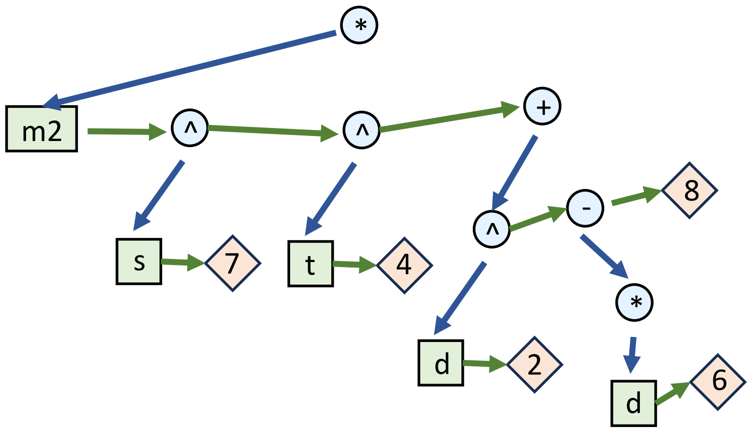
string representing a **mathematical expression**



symbolic trees ...
... through **linked lists**

easily **insert/remove** nodes and perform operations on **sub-trees**

Rational Functions



substitute line parameterization

$$\begin{cases} s(\eta) = s_i + \eta(s_f - s_i) \\ t(\eta) = t_i + \eta(t_f - t_i) \\ \quad \vdots \end{cases}$$

$$\frac{N(\eta)}{D(\eta)} = \frac{a_0 + a_1\eta + a_2\eta^2 + \ldots}{\eta^{m_0}(\eta - \eta_1)^{m_1}(\eta - \eta_2)^{m_2}\ldots}$$

coefficient

root

multiplicity

Rational Functions

Image from "Harry Potter and the Philosopher's Stone" (Warner Bros., 2001).
Used here under fair use for educational and humorous purposes.

$$\frac{N(\eta)}{D(\eta)} = \frac{a_0 + a_1\eta + a_2\eta^2 + \dots}{\eta^{m_0}(\eta - \eta_1)^{m_1}(\eta - \eta_2)^{m_2}\dots}$$

coefficient

root

multiplicity

Rational Functions

$$\frac{N(\eta)}{D(\eta)} = \frac{a_0 + a_1\eta + a_2\eta^2 + \ldots}{\eta^{m_0}(\eta-\eta_1)^{m_1}(\eta-\eta_2)^{m_2}\ldots}$$

coefficient → a_1; root → η_1; multiplicity → m_1

- find the global **list of unique** denominator **roots**
- assign an **integer label** to each root
- represent each denominator with **integer labels** and **multiplicites**
  (fast computation of **LCM**, **simplifications**, **shifts**, etc.)

$\mathbb{N}\ \mathbb{Q}$ $\mathbb{R}$ $\mathbb{C}$
`gmp`, `mpfr`, `mpc`
for **abitrary precision** arithmetic

← functional, fast, open source and well maintained

Solving around Poles

To solve around a pole, transform the DE matrix to **Fuchsian form**

$$\frac{d}{d\eta}\mathbf{I}(\eta) = \mathbb{A}(\eta)\mathbf{I}(\eta) \quad \longrightarrow \quad \frac{d}{d\eta}\widetilde{\mathbf{I}}(\eta) = \widetilde{\mathbb{A}}(\eta)\widetilde{\mathbf{I}}(\eta)$$

$$\begin{cases} \widetilde{\mathbf{I}}(\eta) = \mathbb{T}^{-1}(\eta)\mathbf{I}(\eta) \\ \widetilde{\mathbb{A}}(\eta) = \mathbb{T}^{-1}(\eta)\mathbb{A}(\eta)\mathbb{T}(\eta) - \mathbb{T}^{-1}(\eta)\frac{d}{d\eta}\mathbb{T}(\eta) \end{cases}$$

← **transformation matrix** (of rational functions)

$$\widetilde{\mathbb{A}}(\eta) = \frac{1}{\eta}\sum_{k=0}^{\infty}\widetilde{\mathbb{A}}_k \eta^k$$

Poincaré rank lowered to zero around the pole $\eta = 0$

$$\widetilde{I}(\eta) = \sum_{\lambda\in S}\eta^{\lambda}\sum_{l=0}^{L_\lambda}\sum_{k=0}^{\infty} c_{\lambda,l,k}\log^{l}(\eta)\eta^{k}$$

eingevalues of the leading order $\widetilde{\mathbb{A}}_0$

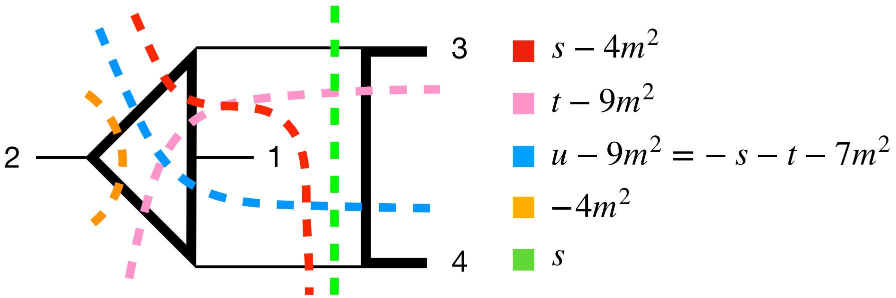
ansatz around **singular** points

Analytic Continuation

Around **branch points:**

$$\widetilde{I}(\eta) = \sum_{\lambda \in S} \eta^{\lambda} \sum_{l=0}^{L_\lambda} \sum_{k=0}^{\infty} c_{\lambda,l,k} \boxed{\log^l(\eta)} \eta^k$$

logarithms give a **branch cut** to the solution!



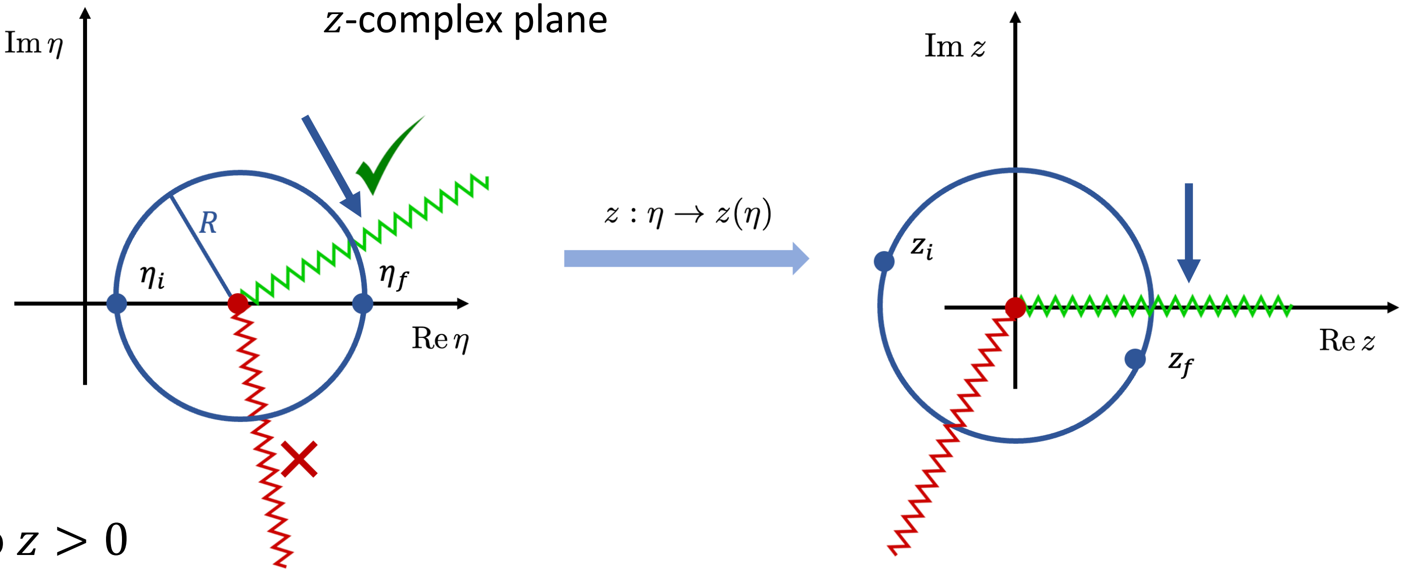
For any given **Cutkosky cut**, consider:

$$z = \underbrace{c_1 s_1 + c_2 s_2 + \cdots}_{\text{invariants flowing through the cut}} - \underbrace{(m_1 + m_2 + \cdots)}_{\text{masses of cut propagators}}{}^2 = z(\eta)$$

map η onto the z-complex plane

Feynman prescription:

$z = 0$ **branch point**
$z > 0$ **branch cut**



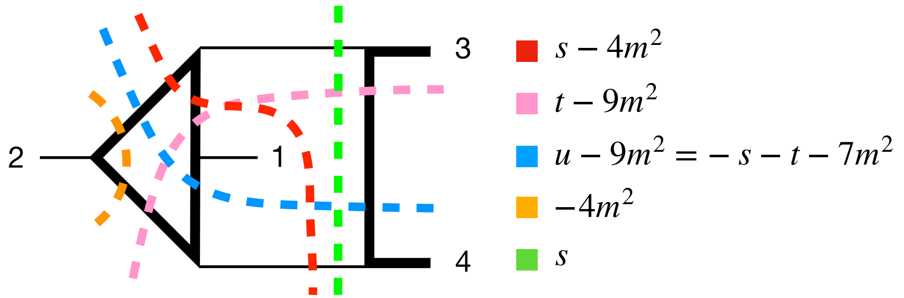
the correct branch-cut in the η-plane is mapped to $z > 0$

Analytic Continuation

Around **branch points:**

$$\widetilde{I}(\eta) = \sum_{\lambda \in S} \eta^{\lambda} \sum_{l=0}^{L_\lambda} \sum_{k=0}^{\infty} c_{\lambda,l,k} \boxed{\log^l(\eta)} \eta^k$$

logarithms give a **branch cut** to the solution!



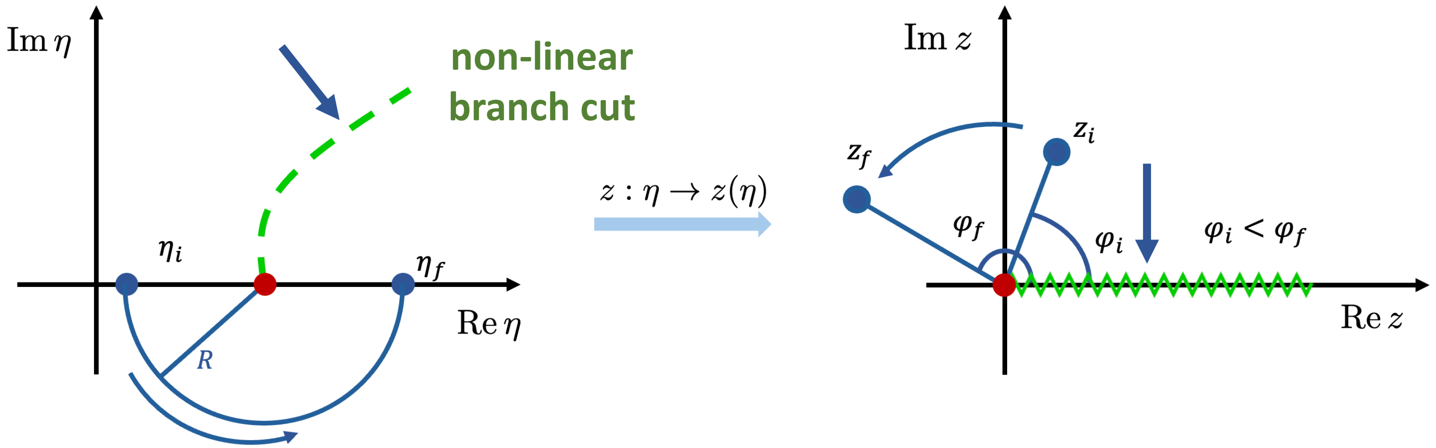
For any given **Cutkosky cut**, consider:

$$z = \underbrace{c_1 s_1 + c_2 s_2 + \cdots}_{\text{invariants flowing through the cut}} - \underbrace{(m_1 + m_2 + \cdots)^2}_{\text{masses of cut propagators}} = z(\eta)$$

map η onto the z-complex plane

Feynman prescription:

$z = 0$ **branch point**
$z > 0$ **branch cut**



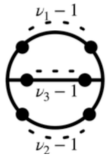
the correct branch-cut in the η-plane is mapped to $z > 0$

Automated Boundary Conditions

AMFlow method

- introduce **auxiliary mass** (no. MIs increases)
- get BCs at infinity (vacuum integrals)
- propagate to zero mass

implemented in **`LINE`** with interface to **`Kira`**
(**`Kira`** → IBPs → DEs w.r.t. auxiliary mass)



$$= (-1)^\nu \left[\frac{\Gamma(\nu_3-2+\epsilon)\Gamma(\nu_1+\nu_2-2+\epsilon)}{\Gamma(\nu_3)\Gamma(\nu_1+\nu_2)} \, {}_4F_3\left(\begin{matrix}2-\epsilon,\nu_1,\nu_2,\nu_1+\nu_2-2+\epsilon\\ \frac{\nu_1+\nu_2}{2},\frac{\nu_1+\nu_2}{2}+\frac{1}{2}, 3-\nu_3-\epsilon\end{matrix};\frac{1}{4}\right) \right.$$

$$+ \frac{\Gamma(2-\nu_3-\epsilon)\Gamma(\nu_1+\nu_3-2+\epsilon)\Gamma(\nu_2+\nu_3-2+\epsilon)\Gamma(\nu+2\epsilon-4)}{\Gamma(\nu_1)\Gamma(\nu_2)\Gamma(2-\epsilon)\Gamma(\nu+\nu_3-4+2\epsilon)}$$

$$\left. \times \, {}_4F_3\left(\begin{matrix}\nu_3,\nu_1+\nu_3-2+\epsilon,\nu_2+\nu_3-2+\epsilon,\nu-4+2\epsilon\\ \nu_3-1+\epsilon,\frac{\nu+\nu_3-4}{2}+\epsilon,\frac{\nu+\nu_3-3}{2}+\epsilon\end{matrix};\frac{1}{4}\right)\right],$$

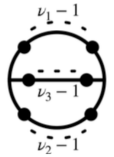
up to **two loops**
(higher-loop planned)

Automated Boundary Conditions

AMFlow method

- introduce **auxiliary mass** (no. MIs increases)
- get BCs at infinity (vacuum integrals)
- propagate to zero mass

implemented in **LINE** with interface to **Kira**
(**Kira** → IBPs → DEs w.r.t. auxiliary mass)



$$=(-1)^{\nu}\left[\frac{\Gamma(\nu_3-2+\epsilon)\Gamma(\nu_1+\nu_2-2+\epsilon)}{\Gamma(\nu_3)\Gamma(\nu_1+\nu_2)}\,{}_4F_3\left(\begin{matrix}2-\epsilon,\nu_1,\nu_2,\nu_1+\nu_2-2+\epsilon\\ \frac{\nu_1+\nu_2}{2},\frac{\nu_1+\nu_2}{2}+\frac{1}{2},3-\nu_3-\epsilon\end{matrix};\frac{1}{4}\right)\right.$$

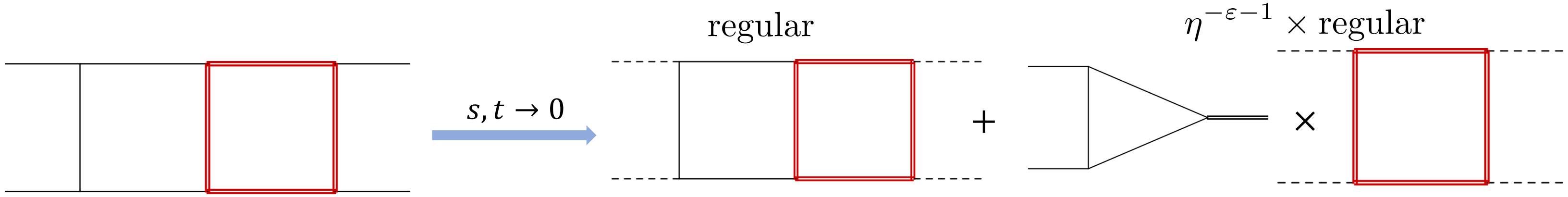
$$+\frac{\Gamma(2-\nu_3-\epsilon)\Gamma(\nu_1+\nu_3-2+\epsilon)\Gamma(\nu_2+\nu_3-2+\epsilon)\Gamma(\nu+2\epsilon-4)}{\Gamma(\nu_1)\Gamma(\nu_2)\Gamma(2-\epsilon)\Gamma(\nu+\nu_3-4+2\epsilon)}$$

$$\left.\times\,{}_4F_3\left(\begin{matrix}\nu_3,\nu_1+\nu_3-2+\epsilon,\nu_2+\nu_3-2+\epsilon,\nu-4+2\epsilon\\ \nu_3-1+\epsilon,\frac{\nu+\nu_3-4}{2}+\epsilon,\frac{\nu+\nu_3-3}{2}+\epsilon\end{matrix};\frac{1}{4}\right)\right],$$

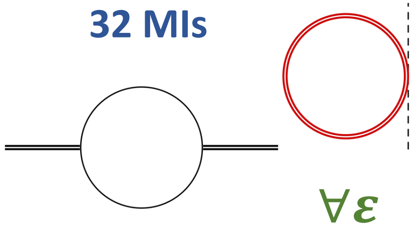
up to **two loops**
(higher-loop planned)

Expansion By Regions

Constraint the solution to have the proper behaviour around singular points **(no additional parameters)**



only 1-loop tadpole and **massless bubble** needed for **32 MIs**



$\forall\varepsilon$

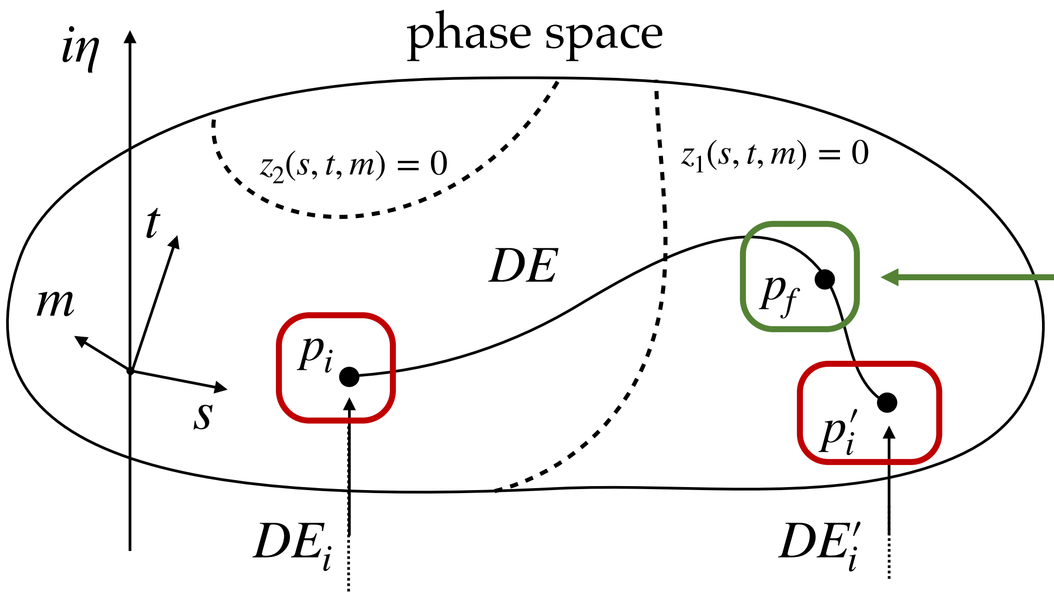
$$\vec{I}(\eta)=c_1\eta^{\lambda_1-n_1}\vec{h}_1(\eta)+c_2\eta^{\lambda_2-n_2}\vec{h}_2(\eta)+\cdots+\vec{p}(\eta)$$

$s,t\propto\eta \qquad \lambda_i\in[0,1[\,,\ n_i\in\mathbb{Z}$

impose **cancellation** of **unwanted power behaviours** → linear relations among coefficients

successful on few examples

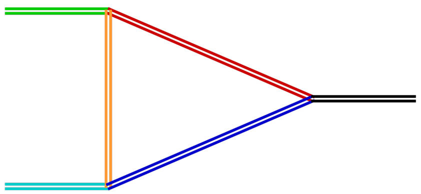
Automated Boundary Conditions

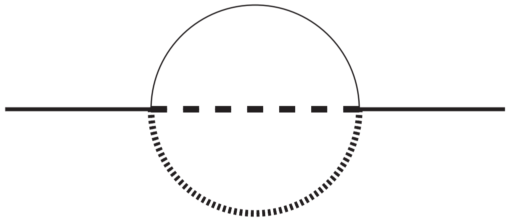


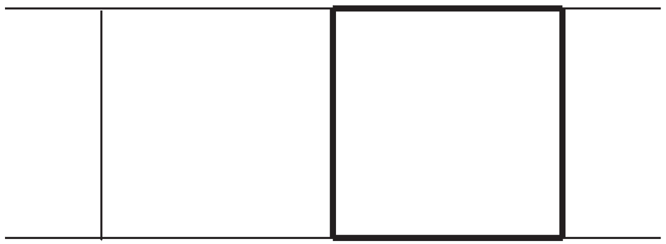
Get to the same point using **different paths** and even **different boundary conditions**

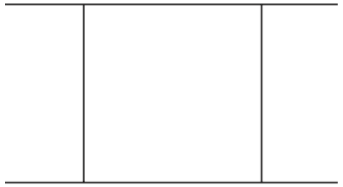
✓ verify internal consistency
✓ **estimate error**

(**no external tools** for integral evaluation)

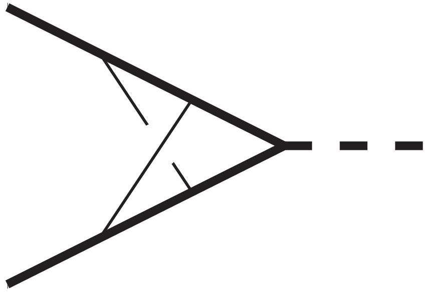


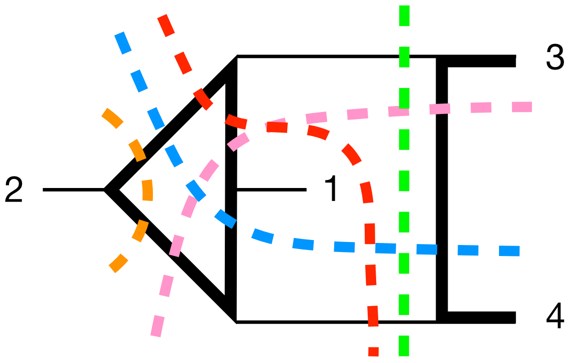


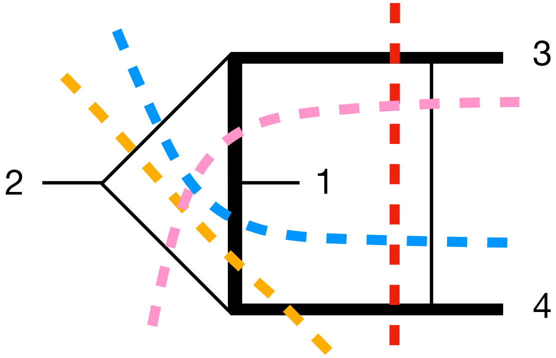




Examples







Running Times

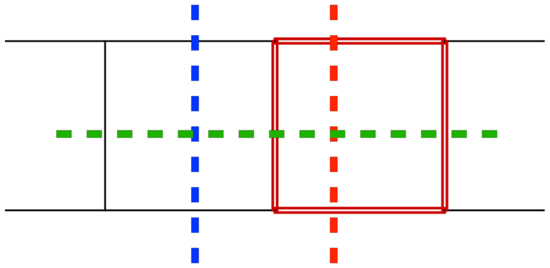
! DISCLAIMER

- a **fair comparison** is **difficult** to perform (no tools that perform both BC computations and propagation)
  - computing BCs with the AMFlow method involves more MIs, an additional mass, IBP reduction
  - phase space propagation is much faster
- let us focus on the **DE solver**
  - AMF^0 vs **`AMFlow`** mathematica package
  - force **`AMFlow`** to use the **"all propagator mode"** → insert the auxiliary mass in **every propagator** (non-recursive) ⇒ we have precisely the **same DEs**
  - **not completely fair**: we use a lower-level language
  - **LINE** was developed paying attention to optimization, but **no direct attempt** to make it faster has been done so far
    - we are aware of several possible improvements, but we expect a speed-up of **no more than a factor 2 or 3**

```
AMD Opteron Processor 4386
10 cores, 64Gb RAM
```

2-Loop Box with a Massive Loop (3 Scales)

n. MI (std-DE): 32
n. MI (η-DE): 68



- $AMF^0 \rightarrow P_1$ (12 reg + 2 sing) :
  - **Kira:** 133s
  - **propagation:**
    - **LINE:** 286s
    - **AMFlow:** 1740s
  - **AMFlow (fully recursive):** 222s
- $EBR(s, t \rightarrow 0) \rightarrow P_1$ (1 sing):
  - **LINE:** 6s
- $P_1 \rightarrow P_2$ (18 reg + 4 sing) :
  - **LINE:** 41s

target	P_1	P_2	P_3
from	AMF^0, EBR ✓ check	P_1	AMF^0, P_2 ✓ check
ϵ^{-4}	0	0	+1.632653061224490e-5
ϵ^{-3}	0	0	-1.507074533571472e-4 +1.025826172600749e-4*i
ϵ^{-2}	-1.684311982263061e-3	+7.121750612221514e-5 +1.223851404355579e-4*i	+2.720746512604996e-4 -9.469228566160803e-4*i
ϵ^{-1}	+4.026956116103587e-3	-7.645333935948279e-4 -3.758110807119310e-4*i	+1.572347464421193e-3 +3.059428585636381e-3*i
ϵ^{0}	-3.997722931454625e-3	+1.621191987913520e-3 -1.376157443003446e-4*i	-8.340803170789194e-3 -2.581654837967916e-3*i
ϵ^{1}	+6.237012138664067e-3	-2.779941041112323e-3 -3.108819053117712e-5*i	+1.483674698459523e-2 -8.593463886823766e-3*i
ϵ^{2}	-4.987777863769356e-3	+5.841649978319638e-3 -1.900890782973601e-3*i	-4.995133665555594e-3 +2.645276326148751e-2*i

```
point: [
  s = -1,
  t = 2,
  m2 = 1
]
```

$s - 4m^2 < 0$
$s - 4m^2 < 0$
$s < 0$

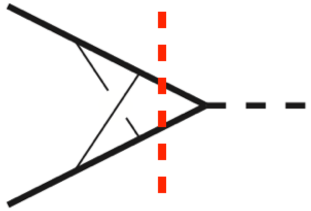
```
point: [
  s = 70,
  t = 50,
  m2 = 10
]
```

$s - 4m^2 > 0$
$s - 4m^2 > 0$
$s > 0$

```
point: [
  s = 70,
  t = 50,
  m2 = 0
]
```

2-Loop Non-Planar Triangle with a Mass (2 Scales)

n. MI (std-DE): 16
n. MI (η-DE): 52



- $AMF^0 \to P_1$ (16 reg + 2 sing) :
  - **Kira:** 28s
  - **propagation:**
    - **LINE:** 210s
    - **AMFlow:** 1200s
  - **AMFlow (fully recursive):** 1500s
- $P_1 \to P_2$ (5 reg + 1 sing) :
  - **LINE:** 4s

target	P_1	P_2	P_3
from	AMF^0	P_1	AMF^0, P_2 ✓ check
ϵ^{-4}	0	0	+1.000000000000000e0
ϵ^{-3}	0	0	-1.154431329803066e0 +6.283185307179586e0*i
ϵ^{-2}	0	0	-2.894245735565264e1 -7.253505969566414e0*i
ϵ^{-1}	+2.532501153536048e-1 +1.376560680870821e-1*i	-3.058450755305179e-2	+6.680132569623135e-1 -9.916741832990889e1*i
ϵ^{0}	-1.137868788629137e0 +1.315450793632957e0*i	+6.882432933483959e-2	+2.306015883275194e2 -9.125506150626736e1*i
ϵ^{1}	-5.535444498587951e0 -1.578608277056101e0*i	+5.232509250247894e-2	+4.317677285401460e2 +3.615355918032282e2*i
ϵ^{2}	-1.199497745643981e1 -8.780073080609521e0*i	+8.195254040212031e-1	+1.850496772277360e1 +1.260787755350661e3*i

```
point: [
  s = 10,
  m2 = 1
]
```

$s - 4m^2 > 0$

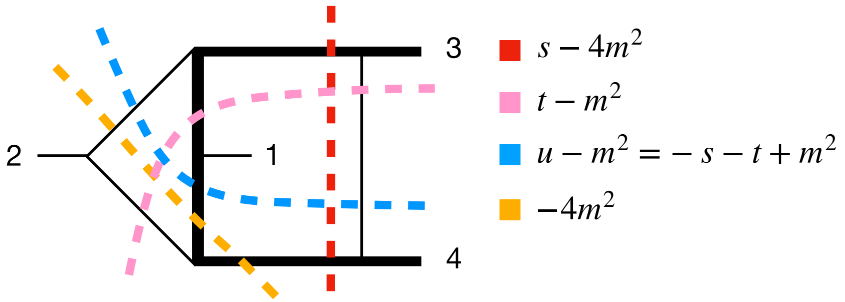
```
point: [
  s = 1,
  m2 = 3
]
```

$s - 4m^2 < 0$

```
point: [
  s = 1,
  m2 = 0
]
```

2-Loop Non-Planar Boxes with a Mass (3 Scales)

n. MI (std-DE): 55
n. MI (η-DE): 144



- $AMF^0 \rightarrow Q_1$ (31 reg + 2 sing) :
  - **Kira:** 15180s
  - **propagation:**
    - **LINE:** 6600s
    - **AMFlow:** 19740s
  - **AMFlow (fully recursive):** 5040s
- $Q_1 \rightarrow Q_2$ (26 reg + 6 sing) :
  - **LINE:** 214s

target	Q_1	Q_2
from	AMF^0	AMF^0, Q_1 ✓ check
ϵ^{-4}	0	0
ϵ^{-3}	-2.634309928357791e-7	+7.825617108436437e-8 -2.554478084014810e-7*i
ϵ^{-2}	+2.177434402618331e-6 -1.655185743641498e-6*i	+5.136099594647812e-9 +3.245051324395477e-6*i
ϵ^{-1}	+2.177434402618331e-6 +1.533076938553119e-5*i	+5.136099594647812e-9 -3.407024087192466e-5*i
ϵ^{0}	-2.810879169233962e-5 -3.761642841819541e-5*i	+2.470711494037188e-4 -6.343358651146831e-5*i
ϵ^{1}	+6.424181660342731e-5 +3.595559671704640e-5*i	+3.561272520516187e-5 +6.872261543040661e-4*i
ϵ^{2}	-1.721862393547420e-4 -1.231788432398794e-5*i	-7.247299398344942e-4 +6.092012063072394e-5*i

$s - 4m^2 < 0$
$t - m^2 < 0$
$-s - t + m^2 > 0$

```
point: [
  s = 1,
  t = 2,
  m2 = 100
]
```

```
point: [
  s = 500,
  t = 150,
  m2 = 100
]
```

$s - 4m^2 > 0$
$t - m^2 > 0$
$-s - t + m^2 < 0$

Scaling of the Performance

Generation of boundary values with **LINE**'s AMF^0

precision digits				
results	decimal	8	16	32
internal	binary	313	506	893

topology	**η-MIs**	**running time (sec)**		
non-planar triangle	52	102	210	531
planar box	68	158	286	762
non-planar box	144	3066	6600	14350

$\Delta t \sim (\#\,\mathrm{MIs})^4$

$\Delta t \sim (\#\,\mathrm{digits})^{1.5}$

scaling dominated by **numerical rather than high-level** operations

AMD Opteron Processor 4386
10 cores, 64Gb RAM

Conclusions and Outlook

Conclusions and Outlook

- `LINE` solves **DEs** to compute **boundary conditions** and **propagate** loop integrals within a **single framework**
- **interface to Kira** for the implementation of the **AMFlow method** (possible extension to more than two loops)
- **automated BCs** with **EBR** (generalization under investigation)
- self-contained evaluation of **numerical accuracy** (no need for external tools to evaluate integrals)
- **open-source** and written primarily in **C** $\rightarrow$ suitable for **computations on clusters**
- code available at https://github.com/line-git/line

Conclusions and Outlook

- **LINE** solves **DEs** to compute **boundary conditions** and **propagate** loop integrals within a **single framework**
- **interface to Kira** for the implementation of the **AMFlow method** (possible extension to more than two loops)
- **automated BCs** with **EBR** (generalization under investigation)
- self-contained evaluation of **numerical accuracy** (no need for external tools to evaluate integrals)
- **open-source** and written primarily in **C** → suitable for **computations on clusters**
- code available at https://github.com/line-git/line

- higher-loop (> 2)
- performance optimization
- high-level coding structure
  - automated generation of unitary cuts
  - smart choice of PS propagation path
  - AMFlow iterative strategy
  - ... and more!

Conclusions and Outlook

- **LINE** solves **DEs** to compute **boundary conditions** and **propagate** loop integrals within a **single framework**
- **interface to Kira** for the implementation of the **AMFlow method** (possible extension to more than two loops)
- **automated BCs** with **EBR** (generalization under investigation)
- self-contained evaluation of **numerical accuracy** (no need for external tools to evaluate integrals)
- **open-source** and written primarily in **C** $\rightarrow$ suitable for **computations on clusters**
- code available at https://github.com/line-git/line

stay tuned!

- higher-loop (> 2)
- performance optimization
- high-level coding structure
  - automated generation of unitary cuts
  - smart choice of PS propagation path
- AMFlow iterative strategy
- ... and more!

Image from "Harry Potter and the Philosopher's Stone" (Warner Bros., 2001).
Used here under fair use for educational and humorous purposes.

Backup

Loop Integrals via Differential Equations

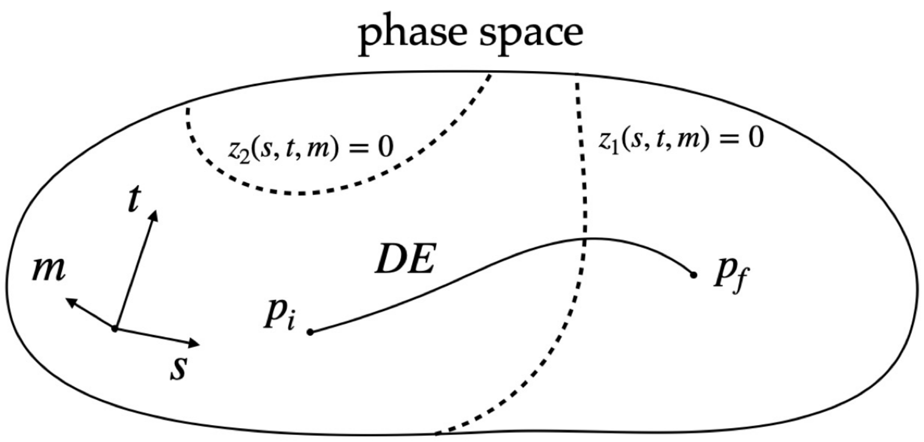
DEs can be used to propagate **Feynman integrals** across the phase space

- use a starting point where **boundary conditions** can be obtained
- find DEs along the line connecting the **initial** and the **target point**
- solve DEs via **series expansion** for different **numerical values of ε**
- **interpolate** ε orders $\quad I = \frac{1}{\epsilon^p} \sum_{k=0}^{\infty} I_k \epsilon^k$

$$\begin{cases} s(\eta) = s_i + \eta(s_f - s_i) \\ t(\eta) = t_i + \eta(t_f - t_i) \\ \vdots \end{cases}$$

$\eta \in [0,1]$
line parameter

$$\partial_\eta = (s_f - s_i)\partial_s + (t_f - t_i)\partial_t + \dots$$
$$A(\eta) = (s_f - s_i)A_s + (t_f - t_i)A_t + \dots$$



starting point $\quad$ *propagation* $\quad$ **target point**

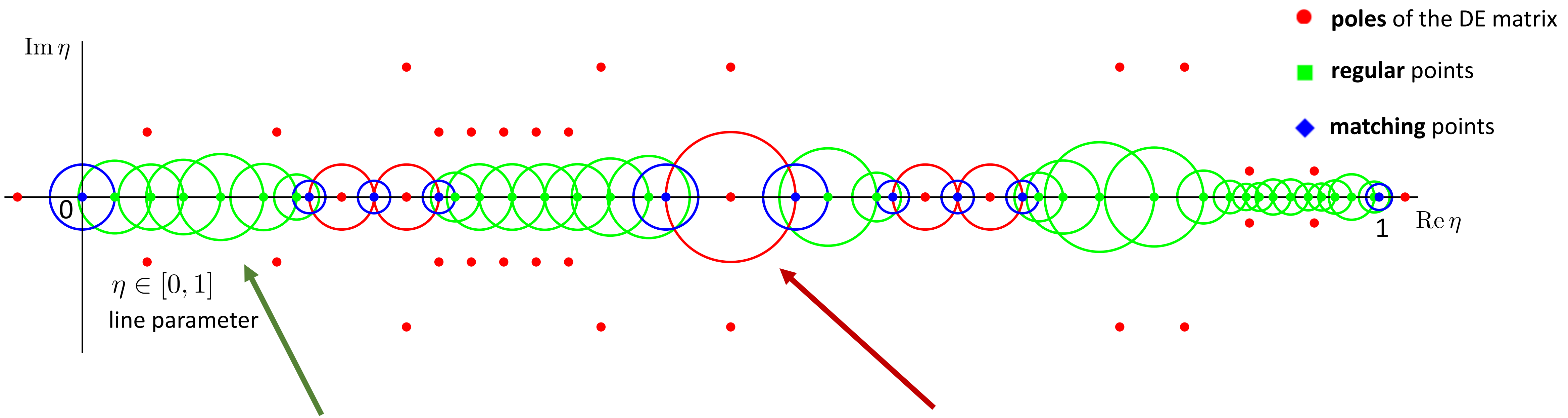
$$\vec{I}(s_i, t_i, \dots) \rightarrow \vec{I}(s_f, t_f, \dots)$$

Master Integrals (MIs): set of independent Feynman integrals

space-time dimension
$d = 4 - 2\epsilon$
(*dimensional regularization*)

Poles and Series Expansion

DEs have **poles** → series expansion within **radius of convergence**



Near regular points the solution has a Taylor expansion:

$$I(\eta) = \sum_{k=0}^{\infty} c_k \eta^k$$

ansatz around **regular** points

Cross singular points using:

max log power

$$I(\eta) = \sum_{\lambda \in S} \eta^\lambda \sum_{l=0}^{L_\lambda} \sum_{k=0}^{\infty} c_{\lambda,l,k} \log^l(\eta) \eta^k$$

set of eigenvalues

ansatz around **regular-singular** points

ε - dependence

DEs depend on the space-time dimension $\mathrm{d} = 4 - 2\varepsilon$ (**dimensional regularization**)

- solve DEs for different **numerical values of epsilon**
- **interpolate** epsilon orders (strategy implemented in the AMFlow package)

trade two-variate problem for multiple **independent univariate** problems of **equal complexity** **(parallelization)**

$$A(\varepsilon, \eta) \to A(\varepsilon_1, \eta), A(\varepsilon_2, \eta), \ldots$$

$$\vec{I} = \frac{1}{\epsilon^p} \sum_{k=0}^{\infty} \vec{I}_k \epsilon^k$$ **Laurent expansion**

$$\varepsilon = \frac{101}{146700}$$

```
MI0: 2.90441164026269689878103960...e3
MI1: 4.35540146043925901671524332...e3
MI2: 7.25644994136370315072624217...e3
MI3: 1.45102408899755166053045532...e3
MI4: 1.45063139101440792324071813...e3
MI5: 1.45053840930241730004028775...e3
MI6: −1.5499533725585654771242419...e−1
```

$$\varepsilon = \frac{17}{24450}$$

```
MI0: 2.87593175196824945061702911...e3
MI1: 4.31268163190383444307933535...e3
MI2: 7.18525024330887846376219687...e3
MI3: 1.43678414841565105694008735...e3
MI4: 1.4363914550165960790255637...e3
MI5: 1.4362984743558711152632678...e3
MI6: −1.5499351276863830613546647...e−1
```

⋮

interpolation →

```
MI5                              bubble
eps^−2: 0
eps^−1: 9.9999999999999999999999e−1
eps^0:  −1.938704283006374112859e0
eps^1:  2.712629538677921993042e0
eps^2:  −3.230718213168225851725e0
eps^3:  3.549244487845649233171e0
eps^4:  −3.732208990036951965128e0
eps^5:  3.832940123529834150758e0

MI6
eps^−2: 0                        triangle
eps^−1: 0
eps^0:  −1.55179783617978100156e−1
eps^1:  2.68153011739011483628e−1
eps^2:  −3.62299367768517539729e−1
eps^3:  4.21339386265382316643e−1
eps^4:  −4.57066908219392659241e−1
eps^5:  4.77147594900988835882e−1
```

Block Strategy

Exploit the **block lower triangular** structure of the DE matrix:

$$A = \begin{pmatrix} A_1 & 0 & 0 & 0 \\ A_3 & A_2 & & 0 \\ & & & 0 \\ A_6 & A_5 & & A_4 \end{pmatrix}$$

- solve one block at a time (much smaller problem)
- trade homogeneous DE for **non-homogeneous** ones

After solving $b-1$ blocks:

$$\partial_\eta \vec{I}_b = A_{b,b}(\eta)\vec{I}_b + \vec{Y}_b$$

solve **non-homogeneous** DEs around **regular-singular** points by **series expansion**

$$\vec{Y}_b = (A_{b,1}, \ldots, A_{b,b-1}) \underbrace{\begin{pmatrix} \vec{I}_1 \\ \ldots \\ \vec{I}_{b-1} \end{pmatrix}}_{\text{known from previous blocks}}$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{D_{11}(\eta)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{D_{11}(\eta)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

For each denominator:

$$-74088\,\eta + 5292\,\eta^2 - 126\,\eta^3 + \eta^4$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{D_{11}(\eta)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**

$$-74088\,\eta + 5292\,\eta^2 - 126\,\eta^3 + \eta^4$$

$$\eta(\eta - 42)^3$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{D_{11}(\eta)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**

{0.00000e0}

global list of unique roots

$$-74088\,\eta + 5292\,\eta^2 - 126\,\eta^3 + \eta^4$$

$$\eta(\eta - 42)^3$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{D_{11}(\eta)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root

```
r0
{0.00000e0}
```

global list of unique roots

$$-74088\,\eta + 5292\,\eta^2 - 126\,\eta^3 + \eta^4$$

$$\eta(\eta - 42)^3$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{\boxed{D_{11}(\eta)}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root

```
    r0          r1
{0.00000e0, 4.2000e1}
```

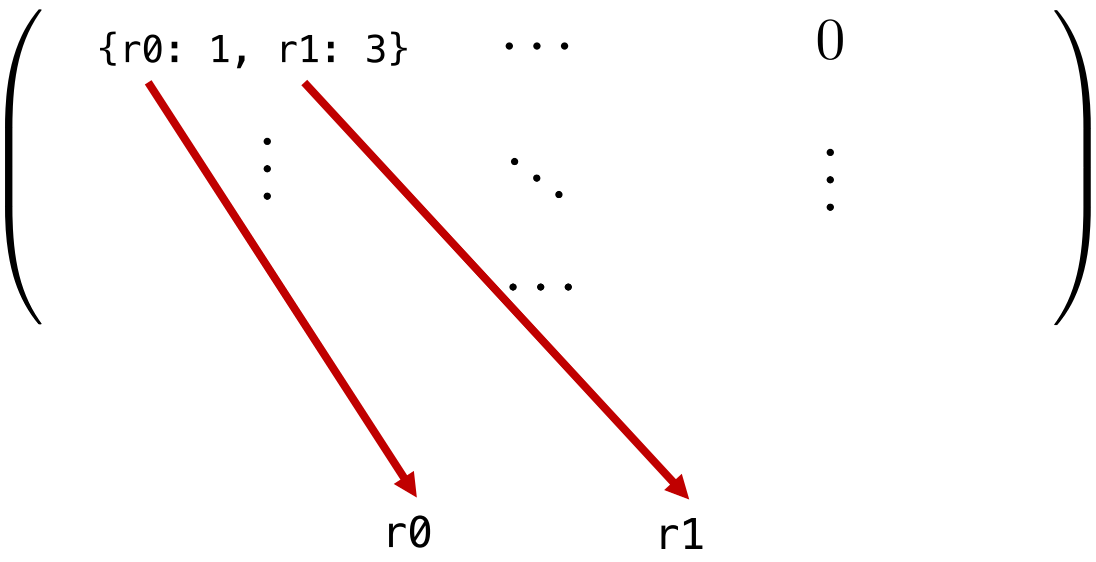
global list of unique roots

$$-74088\,\eta + 5292\,\eta^2 - 126\,\eta^3 + \eta^4$$

$$\eta(\eta - 42)^3$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{D_{11}(\eta)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$



{0.00000e0, 4.2000e1}

global list of unique roots

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

$$-74088\,\eta + 5292\,\eta^2 - 126\,\eta^3 + \eta^4$$

$$\eta(\eta - 42)^3$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

$$\begin{pmatrix} \texttt{\{r0: 1, r1: 3\}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

```
 r0          r1
{0.00000e0, 4.2000e1}
```

global list of unique roots

$$147331044\,\eta^2 - 41681892\,\eta^3 + 4793065\,\eta^4 - 285260\,\eta^5 + 9210\,\eta^6 - 152\,\eta^7 + \eta^8$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

$$\begin{pmatrix} \texttt{\{r0: 1, r1: 3\}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

```
   r0          r1
{0.00000e0, 4.2000e1}
```

global list of unique roots

$$147331044\,\eta^2 - 41681892\,\eta^3 + 4793065\,\eta^4 - 285260\,\eta^5 + 9210\,\eta^6 - 152\,\eta^7 + \eta^8$$

$$\eta^2(\eta-42)^2(\eta-17)^4$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)} \end{pmatrix}$$

$$\begin{pmatrix} \texttt{\{r0: 1, r1: 3\}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

```
r0          r1         r2
{0.00000e0, 4.2000e1, 1.70000e1}
```

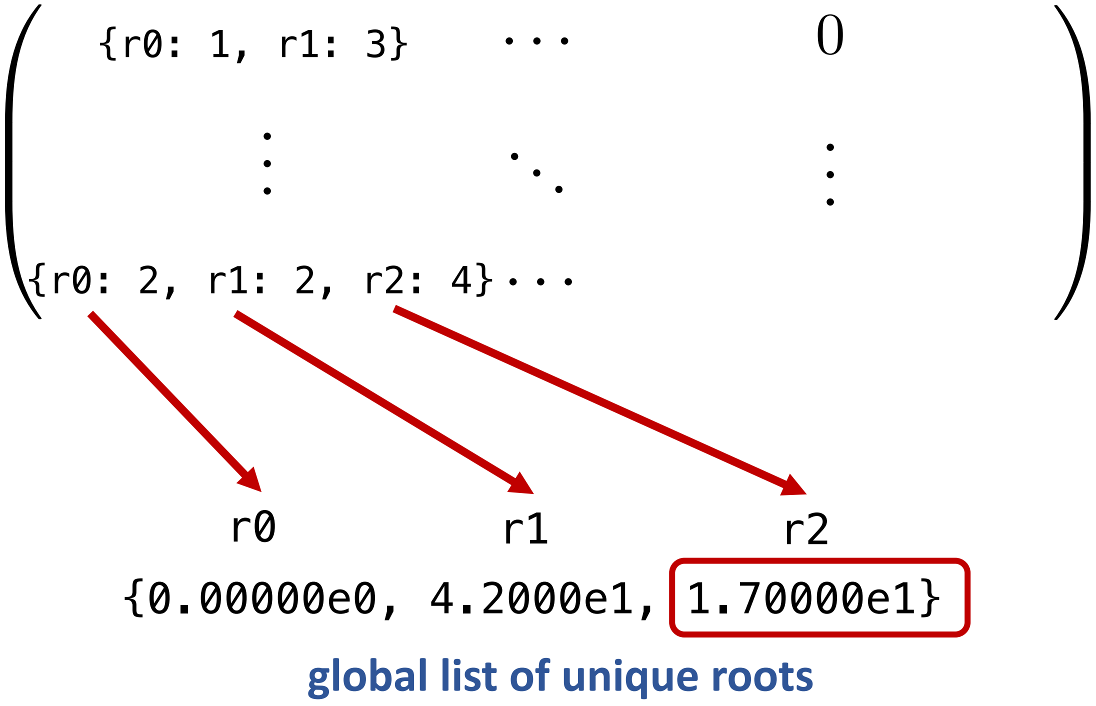
global list of unique roots

$$147331044\,\eta^2 - 41681892\,\eta^3 + 4793065\,\eta^4 - 285260\,\eta^5 + 9210\,\eta^6 - 152\,\eta^7 + \eta^8$$

$$\eta^2(\eta-42)^2(\eta-17)^4$$

Rational Functions

$$
\begin{pmatrix}
\frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
\frac{N_{n1}(\eta)}{D_{n1}(\eta)} & \cdots & \frac{N_{nn}(\eta)}{D_{nn}(\eta)}
\end{pmatrix}
$$



For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

$$147331044\,\eta^2 - 41681892\,\eta^3 + 4793065\,\eta^4 - 285260\,\eta^5 + 9210\,\eta^6 - 152\,\eta^7 + \eta^8$$

$$\eta^2(\eta-42)^2(\eta-17)^4$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{\eta^2(\eta-42)^2(\eta-17)^4} & \cdots & \frac{N_{nn}(\eta)}{D_{n1}(\eta)} \end{pmatrix}$$

$$\begin{pmatrix} \texttt{\{r0: 1, r1: 3\}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \texttt{\{r0: 2, r1: 2, r2: 4\}} & \cdots & \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

```
    r0           r1          r2
{0.00000e0, 4.2000e1, 1.70000e1}
```

global list of unique roots

$$-7408800\,\eta + 2010960\,\eta^2 - 192528\,\eta^3 + 7912\,\eta^4 - 146\,\eta^5 + \eta^6$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{\eta^2(\eta-42)^2(\eta-17)^4} & \cdots & \frac{N_{nn}(\eta)}{\boxed{D_{n1}(\eta)}} \end{pmatrix}$$

$$\begin{pmatrix} \texttt{\{r0: 1, r1: 3\}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \texttt{\{r0: 2, r1: 2, r2: 4\}} & \cdots & \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

```
    r0          r1          r2
{0.00000e0, 4.2000e1, 1.70000e1}
```

global list of unique roots

$$-7408800\,\eta + 2010960\,\eta^2 - 192528\,\eta^3 + 7912\,\eta^4 - 146\,\eta^5 + \eta^6$$

$$\eta(\eta-42)^3(\eta-10)^2$$

Rational Functions

$$\begin{pmatrix} \frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(\eta)}{\eta^2(\eta-42)^2(\eta-17)^4} & \cdots & \frac{N_{nn}(\eta)}{D_{n1}(\eta)} \end{pmatrix}$$

$$\begin{pmatrix} \{r0: 1, r1: 3\} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \{r0: 2, r1: 2, r2: 4\} & \cdots & \end{pmatrix}$$

For each denominator:

- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

r0	r1	r2	r3
{0.00000e0,	4.2000e1,	1.70000e1,	1.00000e1}

global list of unique roots

$$-7408800\,\eta + 2010960\,\eta^2 - 192528\,\eta^3 + 7912\,\eta^4 - 146\,\eta^5 + \eta^6$$

$$\eta(\eta-42)^3(\eta-10)^2$$

Rational Functions

$$
\begin{pmatrix}
\frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
\frac{N_{n1}(\eta)}{\eta^2(\eta-42)^2(\eta-17)^4} & \cdots & \frac{N_{nn}(\eta)}{D_{n1}(\eta)}
\end{pmatrix}
$$

$$
\begin{pmatrix}
\texttt{\{r0: 1, r1: 3\}} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
\texttt{\{r0: 2, r1: 2, r2: 4\}} & \cdots & \texttt{\{r0: 1, r1: 3, r3: 2\}}
\end{pmatrix}
$$

r0	r1	r2	r3
{0.00000e0,	4.2000e1,	1.70000e1,	1.00000e1}

global list of unique roots

For each denominator:

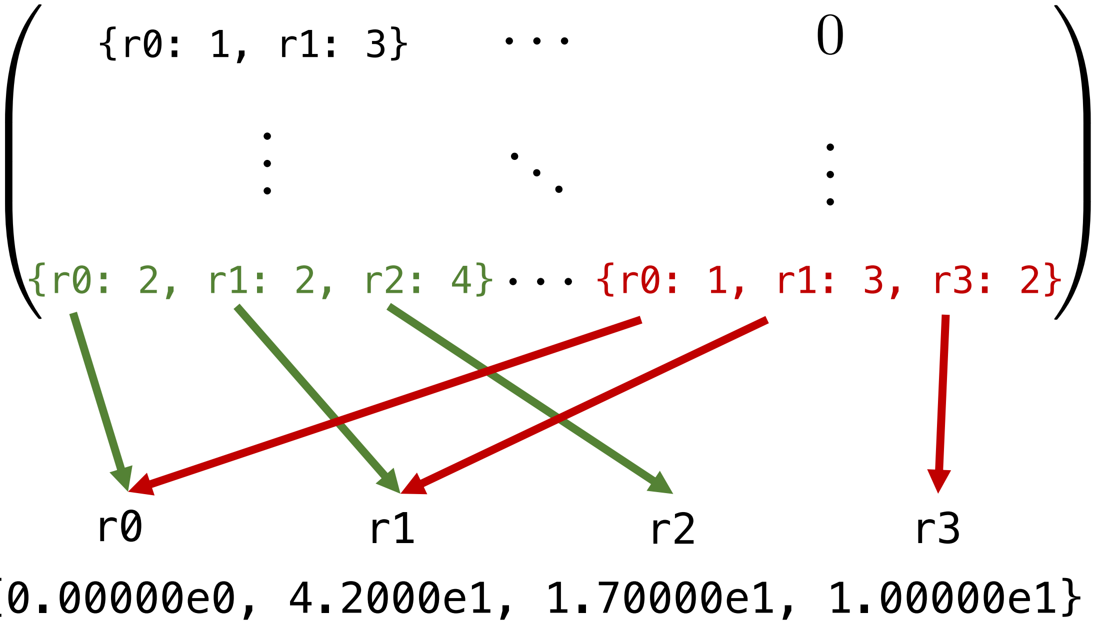
- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

$$-7408800\,\eta + 2010960\,\eta^2 - 192528\,\eta^3 + 7912\,\eta^4 - 146\,\eta^5 + \eta^6$$

$$\eta(\eta-42)^3(\eta-10)^2$$

Rational Functions

$$
\begin{pmatrix}
\frac{N_{11}(\eta)}{\eta(\eta-42)^3} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
\frac{N_{n1}(\eta)}{\eta^2(\eta-42)^2(\eta-17)^4} & \cdots & \frac{N_{nn}(\eta)}{\eta(\eta-42)^3(\eta-10)^2}
\end{pmatrix}
$$



For each denominator:

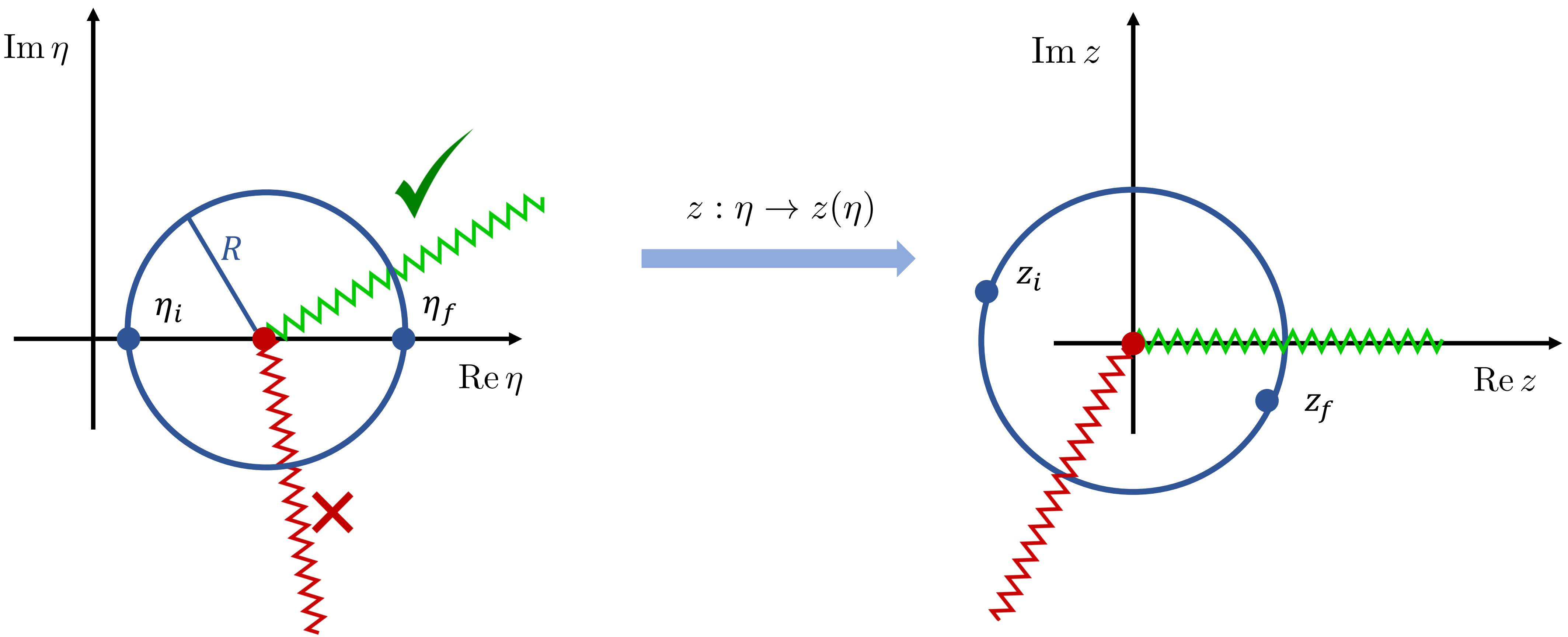
- find roots **numerically** with **arbitrary precision**
- updated a list of **unique roots**
- assign a **label** to each root
- represent each denominator with **integer labels** and **multiplicities**

global list of unique roots
(stored only once and accessed **only when necessary**)

fast computation of polynomial **LCM**, **shifts**, **simplifications**
(only deal with small integers)

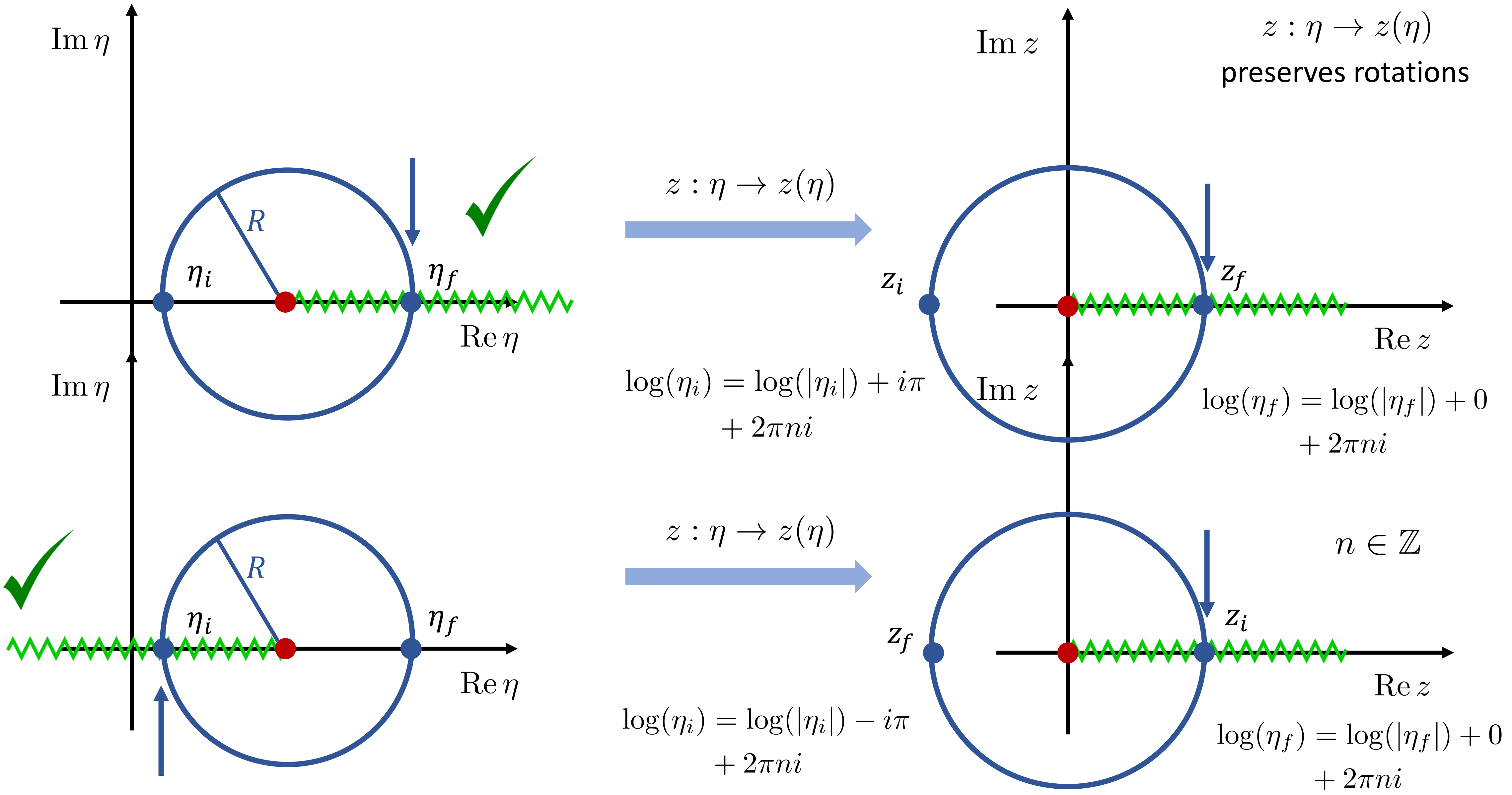
Analytic Continuation – Fixed Masses

The correct branch-cut in the η-plane is mapped to $z > 0$



If cut masses are fixed, the map is linear and there are no complications

Analytic Continuation – Real Fixed Masses



Analytic Continuation – General Case

BUT when **squared masses** are **linearly varied** branch cuts in the η-plane can get **complicated shapes!**

$$z(\eta) = c_1 s_1(\eta) + c_2 s_2(\eta) + \cdots - \left[c_1' \sqrt{m_{1,i}^2 + \eta(m_{1,f}^2 - m_{1,i}^2)} + c_2' \sqrt{m_{2,i}^2 + \eta(m_{2,f}^2 - m_{2,i}^2)} + \cdots\right]^2$$

$$\begin{cases} m_1^2(\eta) = m_{1,i}^2 + \eta(m_{1,f}^2 - m_{1,i}^2) \\ m_2^2(\eta) = m_{2,i}^2 + \eta(m_{2,f}^2 - m_{1,i}^2) \\ \quad\vdots \end{cases}$$

Varying **linear masses** instead:

$$z(\eta) = c_1 s_1(\eta) + c_2 s_2(\eta) + \cdots - [c_1' (m_{1,i} + \eta(m_{1,f} - m_{1,i})) + c_2' (m_{2,i} + \eta(m_{2,f} - m_{2,i})) + \cdots]^2$$

the map is **quadratic** → much **easier to handle**

$$\begin{cases} m_1(\eta) = m_{1,i} + \eta(m_{1,f} - m_{1,i}) \\ m_2(\eta) = m_{2,i} + \eta(m_{2,f} - m_{1,i}) \\ \quad\vdots \end{cases}$$

Analytic Continuation – General Case

Only need to know whether the branch cut is in the **upper** or **lower** half-plane!

Is the branch cut, say, in the lower half-plane?

⇓

Imagine to rotate η counter-clockwise from η_i to η_f:

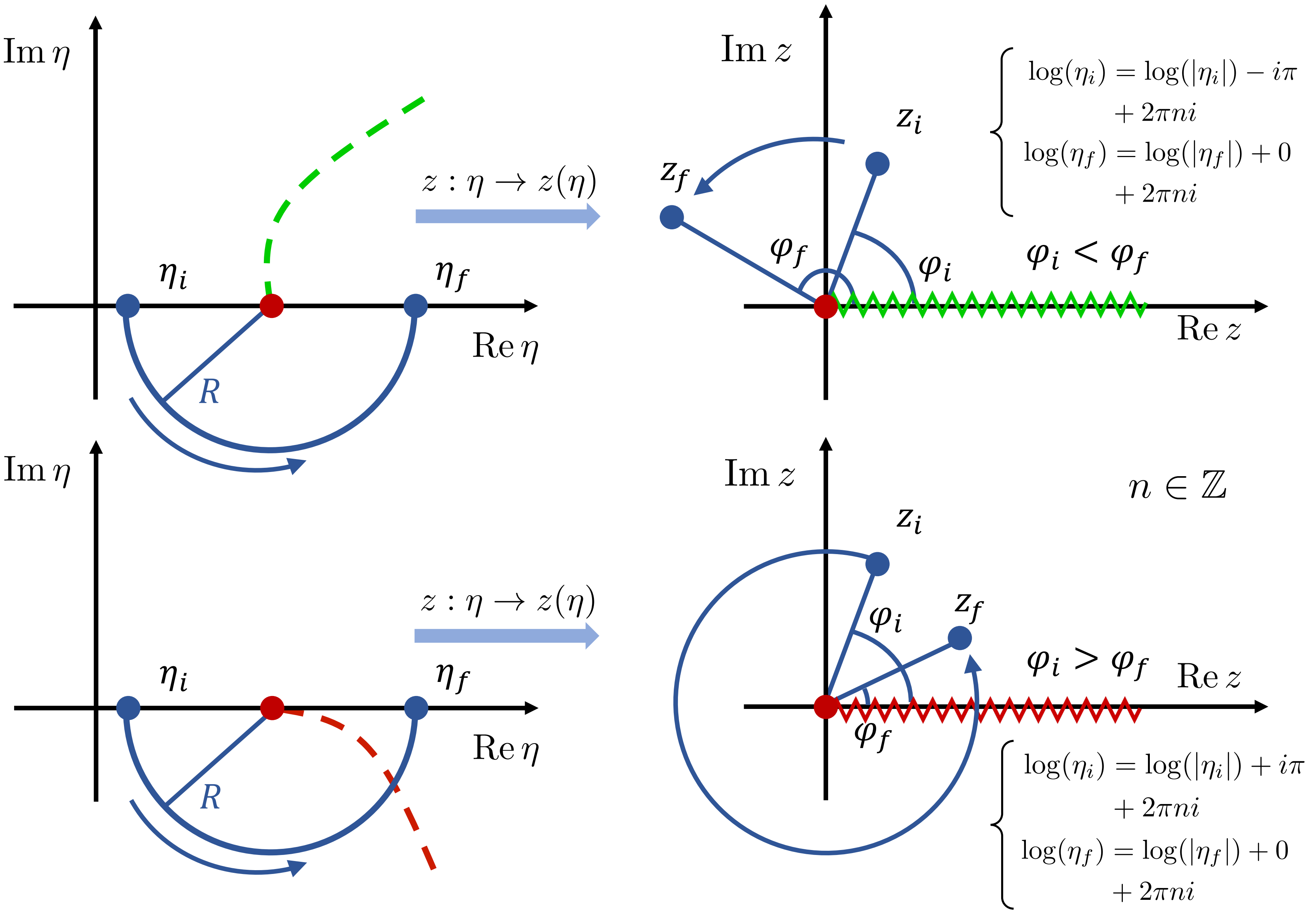
the branch cut is crossed

$\Updownarrow$

z crosses the positive real axis

$\Updownarrow$

$\varphi_i > \varphi_f$



The Auxiliary Mass Flow Method

$AMF^0 \equiv$ the AMFlow method as implmented in **LINE**

- insert an **auxiliary** squared **mass** η in **all denominators**

$$q^2 - m^2 + i0 \quad \longrightarrow \quad q^2 - m^2 - \eta + i0$$

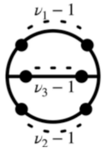
(the number of MIs increases)

- find DEs w.r.t. η (interface to **Kira** to get IBPs)

- compute boundaries for $\eta \to \infty$
(neglect kinematics, **vacuum integrals**)

- propagate from $\eta \to \infty$ to $\eta \to 0$
in the lower complex half-plane
(Feynman prescription is preserved)

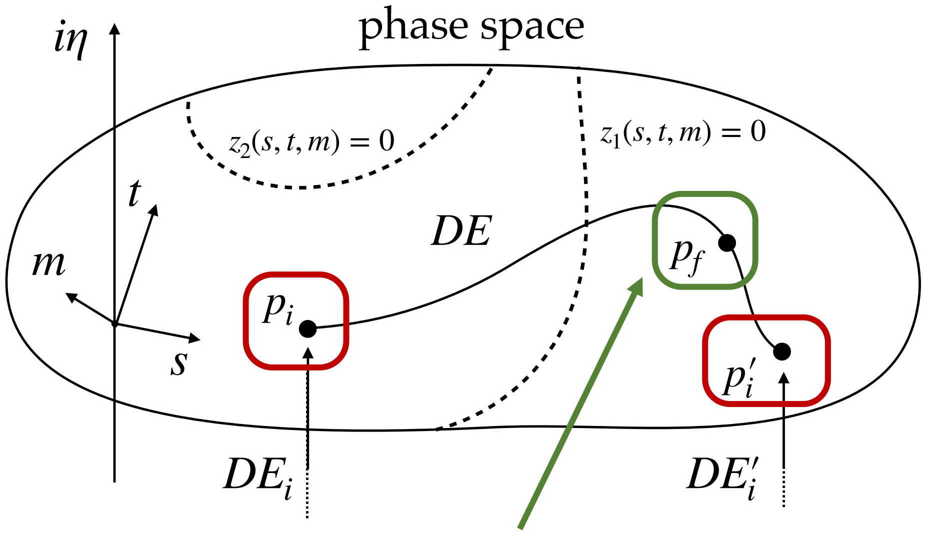
$$= (-1)^\nu \frac{\Gamma(\nu-2+\epsilon)}{\Gamma(\nu)},$$



$$=(-1)^\nu \Bigg[\frac{\Gamma(\nu_3-2+\epsilon)\Gamma(\nu_1+\nu_2-2+\epsilon)}{\Gamma(\nu_3)\Gamma(\nu_1+\nu_2)} \, {}_4F_3\left(\begin{matrix} 2-\epsilon, \nu_1, \nu_2, \nu_1+\nu_2-2+\epsilon \\ \frac{\nu_1+\nu_2}{2}, \frac{\nu_1+\nu_2}{2}+\frac{1}{2}, 3-\nu_3-\epsilon \end{matrix}; \frac{1}{4}\right)$$

$$+ \frac{\Gamma(2-\nu_3-\epsilon)\Gamma(\nu_1+\nu_3-2+\epsilon)\Gamma(\nu_2+\nu_3-2+\epsilon)\Gamma(\nu+2\epsilon-4)}{\Gamma(\nu_1)\Gamma(\nu_2)\Gamma(2-\epsilon)\Gamma(\nu+\nu_3-4+2\epsilon)}$$

$$\times \, {}_4F_3\left(\begin{matrix} \nu_3, \nu_1+\nu_3-2+\epsilon, \nu_2+\nu_3-2+\epsilon, \nu-4+2\epsilon \\ \nu_3-1+\epsilon, \frac{\nu+\nu_3-4}{2}+\epsilon, \frac{\nu+\nu_3-3}{2}+\epsilon \end{matrix}; \frac{1}{4}\right)\Bigg],$$



✓ verify internal consistency
(**no external tools** for integral evaluation)

✓ **estimate error**

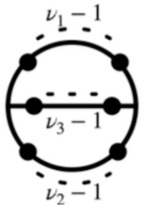
works up to **two loops**
(planned extension to higher-loop)

The Auxiliary Mass Flow Method

- compute boundaries for $\eta \to \infty$
  (neglect kinematics, **vacuum integrals**)

analytic formulas for vacuum integrals only up to two loops

$$= (-1)^{\nu} \frac{\Gamma(\nu-2+\epsilon)}{\Gamma(\nu)} \,,$$



$$= (-1)^{\nu} \Bigg[\frac{\Gamma(\nu_3-2+\epsilon)\Gamma(\nu_1+\nu_2-2+\epsilon)}{\Gamma(\nu_3)\Gamma(\nu_1+\nu_2)} \, {}_4F_3\left(\begin{matrix} 2-\epsilon, \nu_1, \nu_2, \nu_1+\nu_2-2+\epsilon \\ \frac{\nu_1+\nu_2}{2}, \frac{\nu_1+\nu_2}{2}+\frac{1}{2}, 3-\nu_3-\epsilon \end{matrix} ; \frac{1}{4} \right)$$

$$+ \frac{\Gamma(2-\nu_3-\epsilon)\Gamma(\nu_1+\nu_3-2+\epsilon)\Gamma(\nu_2+\nu_3-2+\epsilon)\Gamma(\nu+2\epsilon-4)}{\Gamma(\nu_1)\Gamma(\nu_2)\Gamma(2-\epsilon)\Gamma(\nu+\nu_3-4+2\epsilon)}$$

$$\times \, {}_4F_3\left(\begin{matrix} \nu_3, \nu_1+\nu_3-2+\epsilon, \nu_2+\nu_3-2+\epsilon, \nu-4+2\epsilon \\ \nu_3-1+\epsilon, \frac{\nu+\nu_3-4}{2}+\epsilon, \frac{\nu+\nu_3-3}{2}+\epsilon \end{matrix} ; \frac{1}{4} \right) \Bigg] \,,$$

higher-loop (outlook)

AMFlow **iterative strategy**:

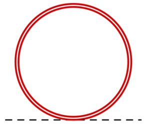
- introduce η in fewer demoniators
- push $\eta \to \infty$
- express boundaries in terms of simpler integrals (**fewer loops**)
- introduce η again in the simpler integrals
- ...

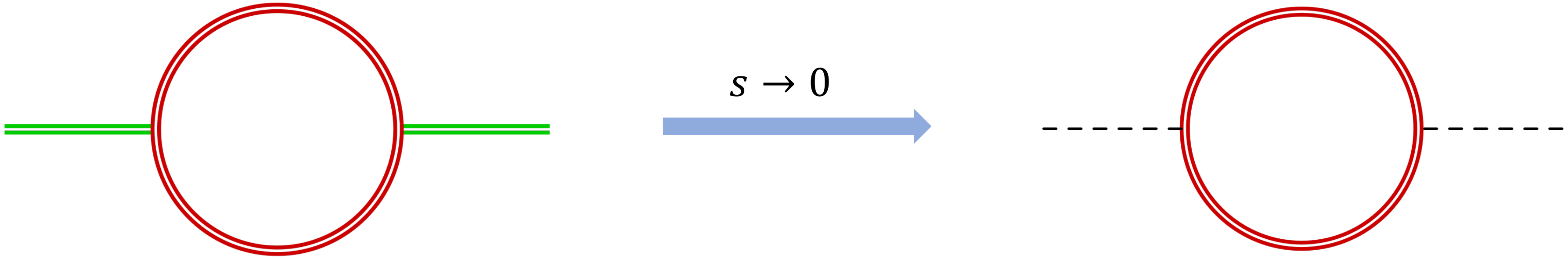
no need for
higher-loop vacuum integrals!

fewer MIs (w.r.t. all-propagator mode)

Boundary Conditions via Expansion by Regions

1-loop massive bubble in the limit of **vanishing kinematics** → only **tadpole** is needed $A = -m^{2(1-\varepsilon)}\Gamma(\varepsilon - 1)$





know **regular** contribution

$$m^{-2\varepsilon}(\varepsilon-1)\Gamma(\varepsilon-1) = -\frac{(\varepsilon-1)}{m^2}A$$

can be obtained from the DE

DE for the bubble

regular

$$\partial_\eta B(\eta) = \frac{f_{B,B}(\eta)}{\eta}B(\eta) + \frac{f_{B,A}(\eta)}{\eta}A \qquad s \propto \eta$$

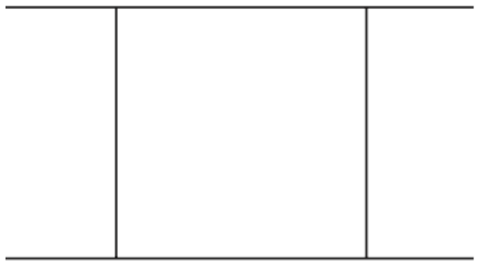
must be regular

$$0 = \frac{f_{B,B}(0)}{\eta}B(0) + \frac{f_{B,A}(0)}{\eta}A \quad \Rightarrow \quad B(0) = -\frac{f_{B,A}(0)}{f_{B,B}(0)}A$$

Boundary Conditions via Expansion by Regions

1-loop massless box in the limit $u \to 0$: only the **bubble** is needed

$$u = 0 \iff s = -t$$



$u \to 0$ regular

DE for the box

regular

s- and t-chunnel bubbles (subsectors)

$$\partial_\eta B(\eta) = \frac{f_{B,B}(\eta)}{\eta} B(\eta) + \frac{f_{B,A}(\eta)}{\eta} A$$

$$u \propto \eta$$

must be regular

$$0 = \frac{f_{B,B}(0)}{\eta} B(0) + \frac{f_{B,A}(0)}{\eta} A \quad \Rightarrow \quad B(0) = -\frac{f_{B,A}(0)}{f_{B,B}(0)} A$$

Boundary Conditions via Expansion by Regions

The implementation is even simpler

solution of the homogeneous

particular solution

$$B(\eta) = c\, H(\eta) + P(\eta)$$

$\eta^{-1/2} \times \text{regular}$

regular

general solution

only $c = 0$ is allowed
to have a regular solution

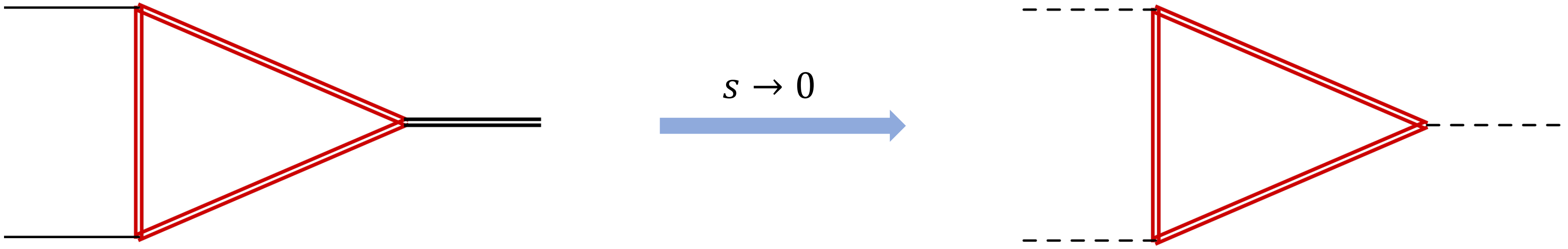
keep just the
particular solution

$$B(\eta) = P(\eta)$$

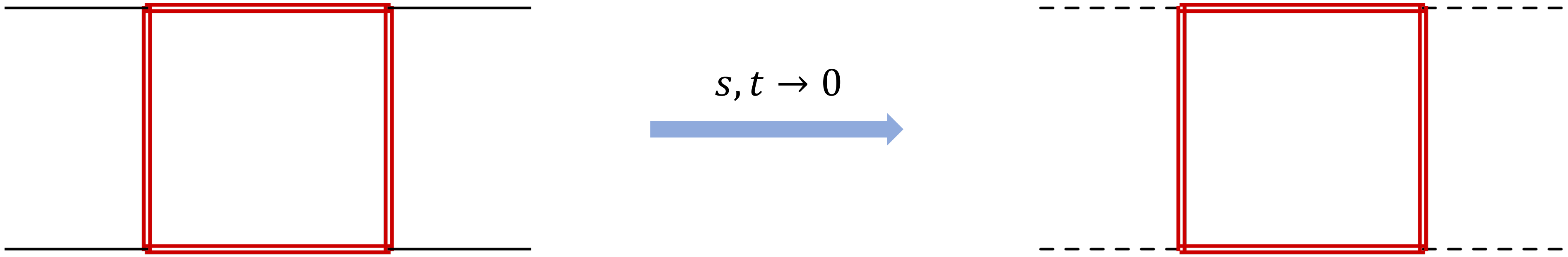
the DE automatically selects, as particular solution, the **unique regular solution**

Boundary Conditions via Expansion by Regions

Analogous for triangle and box



$$-m^{-2(1+\varepsilon)}\frac{\varepsilon}{2}(\varepsilon-1)\Gamma(\varepsilon-1)=\frac{\varepsilon(\varepsilon-1)}{2m^4}A$$

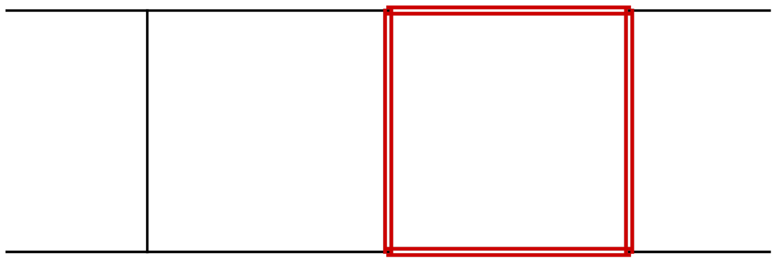


$$m^{-2(2+\varepsilon)}\frac{\varepsilon}{6}(\varepsilon^2-1)\Gamma(\varepsilon-1)=-\frac{\varepsilon(\varepsilon^2-1)}{6m^6}A$$

implementation: keep just the particular solution

Boundary Conditions via Expansion by Regions

A more involved example



loop momenta can be:

- **small** (S) $\rightarrow$ $k_i \ll m$
- **large** (L) $\rightarrow$ $k_i \sim m$

simple rules for propagators

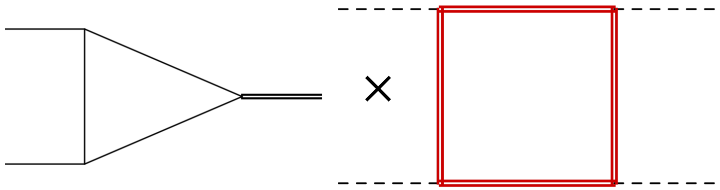
$(k+p)^2 - m^2$ → S: $-m^2$; L: $k^2 - m^2$

$(k+p)^2$ → S: $(k+p)^2$; L: k^2

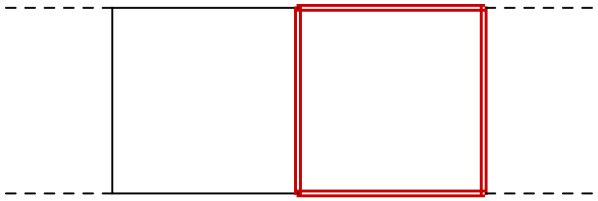
multiple regions for **vanishing external kinematics**:

k_1	k_2	$k_1 + k_2$	
S	S	S	scaleless = 0
S	L	L	→
L	S	L	scaleless = 0
L	L	S	scaleless = 0
L	L	L	→

$\eta^{-\varepsilon-1} \times \text{regular}$



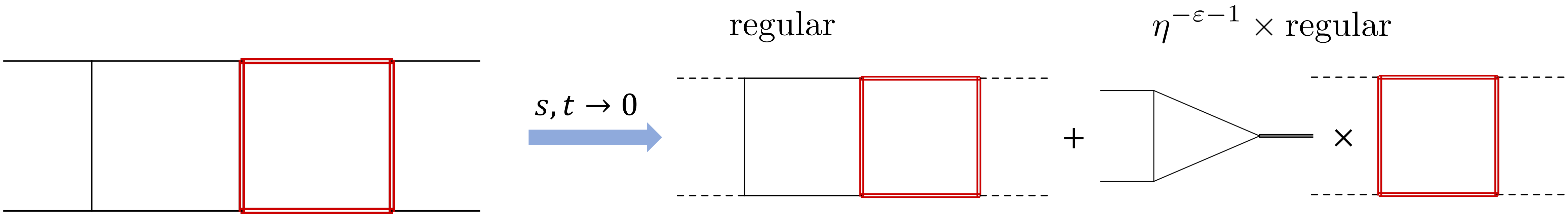
regular



not need to know their expression!

only need the **exponent** of the regions

Boundary Conditions via Expansion by Regions



Solve in a **Fuchsian basis**, then transform back

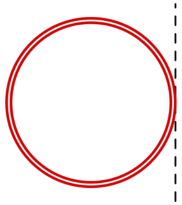
$$\vec{I}(\eta) = c_1 \eta^{\lambda_1 - n_1} \vec{h}_1(\eta) + c_2 \eta^{\lambda_2 - n_2} \vec{h}_2(\eta) + \cdots + \vec{p}(\eta)$$

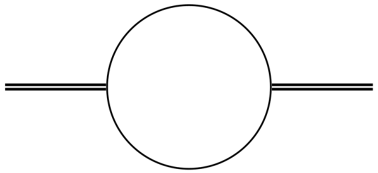
$s, t \propto \eta$

$\lambda_i \in [0, 1[\,, \; n_i \in \mathbb{Z}$

behaviour predicted by **expansion by regions** → impose **cancellation** of **unwanted power behaviours** → linear relations among coefficients

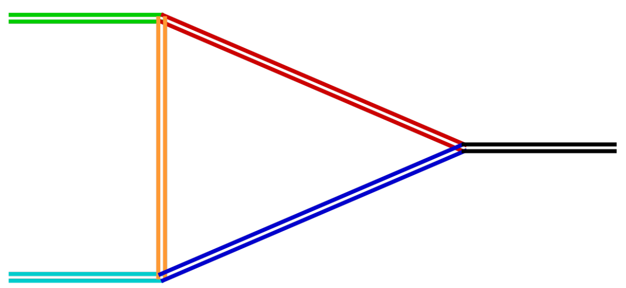
only 1-loop tadpole and **massless bubble** needed for **32 MIs**





$\forall \varepsilon$

1-Loop Triangle with Internal and External Masses



input card written for a specific run

```
order: 5
precision: 16
point: [
  p12 = 1,
  p22 = 2,
  s = 3,
  m12 = 100,
  m22 = 100,
  m32 = 100
]
exit-sing: 1
gen-bound: 1
```

- order → epsilon orders
- precision → precision digits
- point → target point
- exit-sing → start from designated singular point
- gen-bound → automated BC generation

common files written once per topology

```
common/
├── vars.txt
├── 0.txt
├── 1.txt
├── 2.txt
├── 3.txt
├── 4.txt
├── 5.txt
├── branch_cuts.txt
├── initial_point.txt
├── bound_behav.txt
└── bound_build.txt
```

- vars.txt → list of variables: $p_1^2, p_2^2, s, m_1^2, m_2^2, m_3^2$
- 0.txt – 5.txt → DE matrices (one file per variable)
- branch_cuts.txt → list of branch cuts

```
tot-branch: 3
massless-branch: 0
branches: [
  s-(m1+m3)^2,
  p12-(m1+m2)^2,
  p22-(m2+m3)^2
]
```

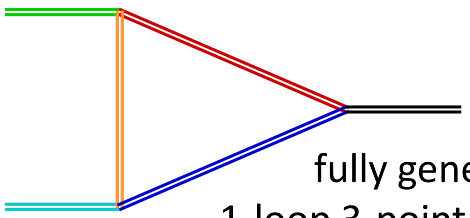
- initial_point.txt → singular point where automated BCs are generated

```
point: [
  s = 0,
  p12 = 0,
  p22 = 0,
  m12 = 100,
  m22 = 100,
  m32 = 100
]
```

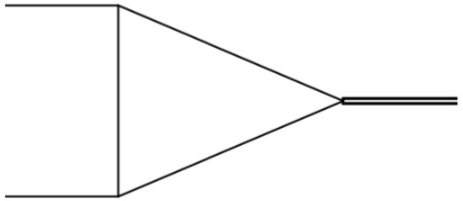
- bound_behav.txt → Expansion by Region exponents (all MIs are regular)
- bound_build.txt → first three MIs are tadpoles

1-Loop Triangle with 6 Scales

n. MI (std-DE): 7
n. MI (η-DE): 7



fully general
1-loop 3-point function
at all orders in ε



$$\frac{1}{\epsilon^2}\frac{\Gamma(1+\epsilon)\Gamma^2(1-\epsilon)}{\Gamma(1-2\epsilon)}\frac{(-s)^{-\epsilon}}{s}$$

target	P_1	P_2	P_3	P_4
from	AMF^0, EBR ✓ check	AMF^0, P_1 ✓ check	P_1	AMF^0, P_3 ✓ check
ϵ^{-2}	0	0	0	-1.000000000000000e0
ϵ^{-1}	0	0	0	+5.772156649015329e-1
ϵ^{0}	-7.599624851460716e-2 -1.024202715501841e-1*i	-5.114624184386078e-2	-9.105983456552547e-2 -3.405963008295366e-2*i	+6.558780715202539e-1
ϵ^{1}	+2.851448508579519e-1 +1.498241156232269e-1*i	+1.461267744725764e-1	+2.054866656214297e-1 +2.780936409230585e-2*i	+2.362111171285093e0
ϵ^{2}	-4.359339557414683e-1 -7.119426049903811e-2*i	-2.508159227043435e-1	-3.033284294289876e-1 -2.327298560596528e-2*i	+1.692738940537638e0
ϵ^{3}	+4.673966245020759e-1 +5.243128182287680e-3*i	+3.394894906445344e-1	+3.792260921703711e-1 +1.589606675868420e-2*i	+2.728361494345973e0
ϵ^{4}	-4.703087868710451e-1 +4.807793030293406e-3*i	-4.033919909274164e-1	-4.294046913943785e-1 -9.903139892953955e-3*i	+1.673348221588670e0

```
point: [
  s = 50,
  m12 = 5,
  m22 = 7,
  m32 = 10,
  p12 = 2,
  p22 = -1/3
]
```

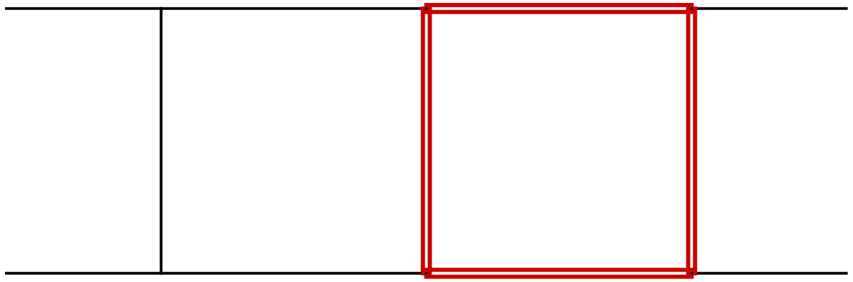
```
point: [
  s = 1,
  m12 = 10,
  m22 = 10,
  m32 = 10,
  p12 = 2,
  p22 = -1/3
]
```

```
point: [
  s = 1,
  m12 = (1 -1),
  m22 = (8/3 -2),
  m32 = (17 -1/4),
  p12 = 2,
  p22 = -1/3
]
```

```
point: [
  s = -1,
  m12 = 0,
  m22 = 0,
  m32 = 0,
  p12 = 0,
  p22 = 0
]
```

$s-(m_1+m_2)^2>0$ $\quad s-(m_1+m_2)^2<0$

2-Loop Box with a Massive Loop (3 Scales)



input card written for a specific run

```
order: 5
precision: 16
point: [
  s = 1,
  t = 2,
  m2 = 100
]
exit-sing: 1
gen-bound: 1
```

- order → epsilon orders
- precision → precision digits
- point → target point
- exit-sing → start from designated singular point
- gen-bound → automated BC generation

common files written once per topology

```
common/
├── vars.txt
├── 0.txt
├── 1.txt
├── 2.txt
├── branch_cuts.txt
├── initial_point.txt
├── bound_behav.txt
└── bound_build.txt
```

- vars.txt → list of variables: s, t, m^2
- 0.txt, 1.txt, 2.txt → DE matrices
- branch_cuts.txt → list of branch cuts

```
tot-branch: 3
massless-branch: 1
branches: [
  s,
  s-4*m^2,
  t-4*m^2
]
```

- initial_point.txt → singular point where automated BC are obtained

```
point: [
  s = 0,
  t = 0,
  m2 = 100
]
```

- bound_behav.txt → Expansion by Region exponents
- bound_build.txt →
  - 1^{st} MI: tadpole squared
  - 2^{nd} MI: tadpole times massless bubble

2-Loop Box with a Massive Loop (3 Scales)

```
point: [
  s = 0,
  t = 0,
  m2 = 100
]
```

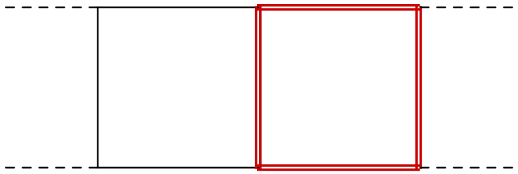
exit sing

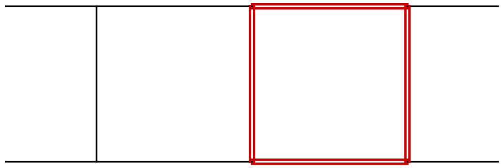
automated BCs

```
point: [
  s = 1,
  t = 2,
  m2 = 100
]
exit-sing: 1
gen-bound: 1
```

propagation *(62 points, 6 singular)*

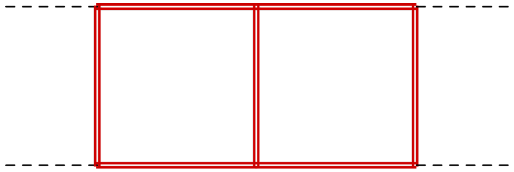
```
point: [
  s = -63845/42,
  t = 1000/11,
  m2 = 100
]
```





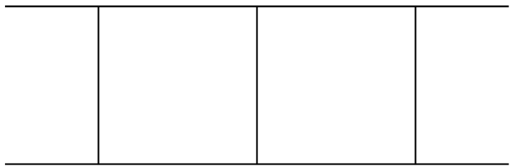
set mass to zero

2-loop **massless** box +
auxiliary complex mass



AMFlow *method*

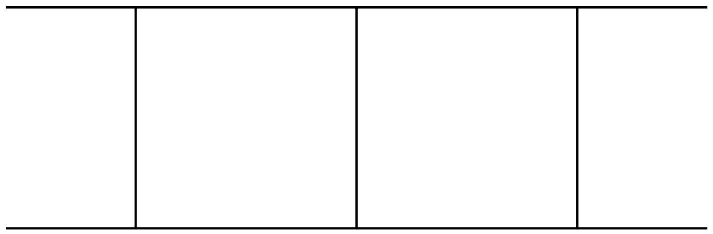
```
point: [
  s = 1,
  t = 2,
  m2 = 0
]
```



consistency check

```
eps^-4: ( 1.9999999999999999999e0   0)
eps^-3: (-4.0417306110059947159 6e0  1.2566370614359172953850e1)
eps^-2: (-4.8001781343062398206 4e1 -2.5394942390650838649408e1)
eps^-1: ( 8.4986796806677272215 9e1 -1.3623727829157781665685e2)
eps^0:  ( 3.2601447892030669435 9e2  1.9980374608773398854703e2)
eps^1:  (-3.5013100968814964513 9e2  6.9082665812695745267367e2)
eps^2:  (-1.3057099661744081740 1e3 -4.5016326405271473861717e2)
eps^3:  ( 4.6485844434139643715 0e2 -2.1952002572148880782472e3)
eps^4:  ( 4.1832603240150610982 6e3  6.1002504659800610703251e2)
eps^5:  ( 2.4973861545306198176 0e3  1.0042259081313034038928e4)
```

2-Loop Massless Box with AMFlow Method



input card:

```
order: 5
precision: 16
point: [
       s = 1,
       t = 2,
]
exit-sing: -1
```

use AMFlow method

needed for **interface to Kira**

```
external-momenta: [p1, p2, p3, p4,]
momentum-conservation: [p4, -p1-p2-p3]
masses: []
loop-momenta: [k1, k2,]
isp: 2
propagators: [
  [k1, 0],
  [k1-p1, 0],
  [k1+p2, 0],
  [k2-p2, 0],
  [k2+p1, 0],
  [k1+k2, 0],
  [k2-p2-p3, 0],
  [k2, 0 ],
  [k1 + p3, 0]
]
kinematic-invariants: [s, t]
squared-momenta: [
  [p1, 0],
  [p2, 0],
  [p3, 0],
  [p1+p2, s],
  [p2+p3, t],
  [p1+p3, -s-t]
]
```

two additional **common files**:

```
common/
├── ...
├── ...
├── ...
├── topology.txt
└── MIs.txt
```

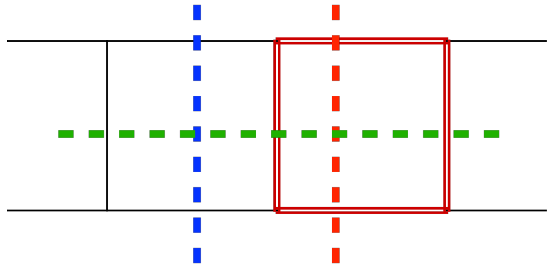
list of master integrals (for eta-less propagation)

```
MI[0, 1, 0, 1, 0, 1, 0, 0, 0]
MI[1, 0, 0, 0, 0, 1, 1, 0, 0]
MI[0, 1, 1, 1, 1, 0, 0, 0, 0]
MI[1, 0, 0, 1, 1, 1, 0, 0, 0]
MI[1, 1, 1, 0, 0, 1, 1, 0, 0]
MI[1, 1, 0, 1, 0, 1, 1, 0, 0]
MI[1, 1, 1, 1, 1, 1, 1, -1, 0]
MI[1, 1, 1, 1, 1, 1, 1, 0, 0]
```

(or just one target integral)

2-Loop Box with a Massive Loop (3 Scales)

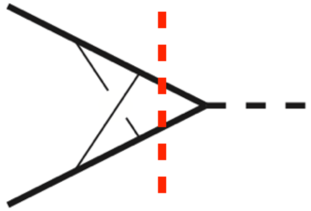
n. MI (std-DE): 32
n. MI (η-DE): 68



	8 digits	16 digits	32 digits
• $AMF^0 \rightarrow P_1$ (12 reg + 2 sing) :	**8 digits**	**16 digits**	**32 digits**
• **Kira:**	133s	133s	133s
• **propagation:**			
• **LINE:**	158s	286s	762s
• **AMFlow:**	1121s	1740s	2827s
• $EBR(s, t \rightarrow 0) \rightarrow P_1$ (1 sing):			
• **LINE:**	4s	6s	22s
• $P_1 \rightarrow P_2$ (18 reg + 4 sing) :			
• **LINE:**	23s	41s	101s

2-Loop Non-Planar Triangle with a Mass (2 Scales)

n. MI (std-DE): 16
n. MI (η-DE): 52

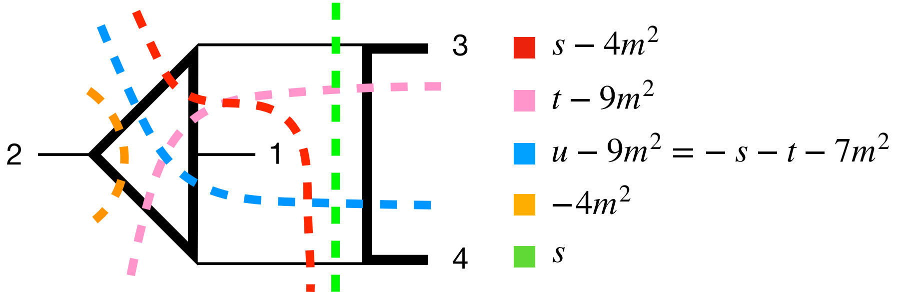


	8 digits	16 digits	32 digits
• $AMF^0 \rightarrow P_1$ (16 reg + 2 sing) :			
• **Kira:**	28s	28s	28s
• **propagation:**			
• **LINE:**	102s	210s	531s
• **AMFlow:**	1087s	1200s	1597s
• $P_1 \rightarrow P_2$ (5 reg + 1 sing) :			
• **LINE:**	2s	4s	8.5s

2-Loop Non-Planar Boxes with a Mass (3 Scales)

n. MI (std-DE): 54

n. MI (η-DE): 89



target	P_1	P_2	P_3	P_4	P_5	P_6
from	AMF^0	AMF^0, P_1 ✓ check	AMF^0	AMF^0, P_3 ✓ check	AMF^0	AMF^0, P_5 ✓ check
ϵ^0	+2.576938753803745e-1 -2.465521721983634e-1*i	+2.518740723653660e-1 -1.169079848124980e-1*i	+2.751593454707949e-1 -3.815281539209958e-1*i	+2.506591535092400e-1 -4.235680397875819e-1*i	-2.405260844173886e-1 -5.984661196233730e-3*i	-4.831181490833649e-1
ϵ^1	+9.839059948409147e-1 -1.447010196851563e-1*i	+8.377932210850515e-1 +2.609108724913395e-1*i	+1.257054227433279e0 +4.342974425182124e-1*i	+1.187415013371159e0 -5.997939132016630e-1*i	-5.588054474320729e-1 -3.250774673987693e-2*i	-1.396083737425863e0
ϵ^2	+1.881565035678200e0 -4.606206236766448e-3*i	+1.544162064068738e0 +1.125263466661532e0*i	+2.478546160626464e0 -2.47049854421279e-1*i	+2.372121269639779e0 -5.961585949177441e-1*i	-1.124284189077083e0 -8.467242364778369e-2*i	-3.146872480560270e0

```
point: [
  s = 3,
  t = 2,
  m2 = 1
]
```

$s - 4m^2 < 0$

```
point: [
  s = 5,
  t = 2,
  m2 = 1
]
```

$s - 4m^2 > 0$

```
point: [
  s = 2,
  t = 8,
  m2 = 1
]
```

$t - 9m^2 < 0$

```
point: [
  s = 2,
  t = 10,
  m2 = 1
]
```

$t - 9m^2 > 0$

```
point: [
  s = -3,
  t = -5,
  m2 = 1
]
```

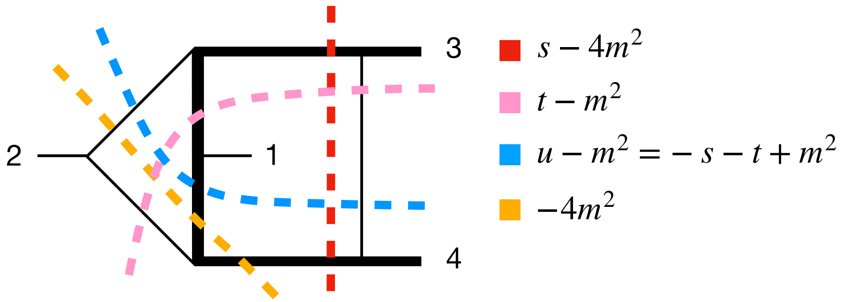
$-s - t - 7m^2 > 0$

```
point: [
  s = -1,
  t = -3,
  m2 = 1
]
```

$-s - t - 7m^2 < 0$

2-Loop Non-Planar Boxes with a Mass (3 Scales)

n. MI (std-DE): 55
n. MI (η-DE): 144



• $AMF^0 \rightarrow Q_1$ (31 reg + 2 sing) :	**8 digits**	**16 digits**	**32 digits**
• **Kira:**	15180s	15180s	15180s
• **propagation:**			
• **LINE:**	3066s	6600s	14350s
• **AMFlow (fully recursive):**			
• $Q_1 \rightarrow Q_2$ (26 reg + 6 sing) :			
• **LINE:**	108s	214s	498s

Higher-Loop (> 2)

- **phase-space propagation (with external BCs):** already there in principle

- **automated BCs with EBR (#loops $\geq$ 2):** generalization under investigation

- **LINE implementation of AMFlow method:**

    analytic formulas for vacuum integrals only up to two loops

    implementation of the AMFlow iterative strategy planned

Jordan Normal Form

Not all $N\times N$ matrices are diagonalizable → **defective matrices**: fewer than N eigenvectors

Jordan normal form:

Jordan blocks

$$A_0' = \begin{pmatrix} \lambda_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_2 \end{pmatrix}$$

heads of the chains

Jordan chain:

$$A_0' \vec{v_1} = \lambda \vec{v_1}$$ head of the chain (eigenvector)

$$A_0' \vec{v_i} = \lambda \vec{v_i} + \vec{v}_{i-1} \quad i = 2, \ldots, L$$

generalized eigenvectors

defective eigenvalues: g.m. $\neq$ a.m.

λ_1: g.m. = 2, a.m. = 6

λ_2: g.m. = 1, a.m. = 3

ill-defined numerical problem:
the structure of the output is affected by the **round-off error**!

$$\lambda_1 \stackrel{?}{=} \lambda_2$$

it depends on the precision!