Preprint typeset in JHEP style - HYPER VERSION

P2Scattering.m v1.6

Bruno Le Floch and Boris Pioline

ABSTRACT: The MATHEMATICA package `P2Scattering.m`, first released alongside our work with Pierrick Bousseau and Pierre Descombes [1], provides a suite of routines to study the scattering diagram for the derived category $\mathcal{C} = D^b(K_{\mathbb{P}^2})$ of coherent sheaves on local $\mathbb{P}^2$ along the large volume, orbifold and Π-stability slices in the space of Bridgeland stability conditions, and extracting the corresponding generalized Donaldson-Thomas invariants $\Omega_Z(\gamma)$. We provide a list of all routines and give a few examples. The current version includes some improvements developed in the course of [2].

Contents

1. Summary

The MATHEMATICA package `P2Scattering.m` provides a suite of routines to study the scattering diagram for the derived category $\mathcal{C} = D^b(K_{\mathbb{P}^2})$ of coherent sheaves on local $\mathbb{P}^2$ along the large volume, orbifold and Π-stability slices in the space of Bridgeland stability conditions, and extracting the corresponding generalized Donaldson-Thomas invariants $\Omega_Z(\gamma)$.

Following [3], the charge vector γ stands either for the Chern vector $[r, d, \mathrm{ch}_2]$, its integral cousin $[r, d, \chi)$ where $\chi = r + \frac{3}{2}d + \mathrm{ch}_2$, or the dimension vector (n_1, n_2, n_3) associated to the tilting sequence

$$(E_1 = i_*(\mathcal{O}), \quad E_2 = i_*(\Omega(1))[1], \quad E_3 = i_*(\mathcal{O}(-1))[2]) \tag{1.1}$$

such that

$$(n_1, n_2, n_3) = (-\frac{3}{2}d - \mathrm{ch}_2 - r, -\frac{1}{2}d - \mathrm{ch}_2, \frac{1}{2}d - \mathrm{ch}_2) = (-\chi, r + d - \chi, r + 2d - \chi) \tag{1.2}$$

The central charge is given by $Z(\gamma) = -rT + dT_D - \mathrm{ch}_2$, where (T, T_D) are

$$T = s + \mathrm{i}t \ , \quad T_D = \frac{1}{2}(s + \mathrm{i}t)^2 \tag{1.3}$$

for the large volume slice, or

$$\begin{pmatrix} T \\ T_D \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{3} \end{pmatrix} + \int_{\tau_o}^{\tau} \begin{pmatrix} 1 \\ u \end{pmatrix} \ C(u) du \tag{1.4}$$

for the Π-stability slice, where $C(\tau) = \frac{\eta(\tau)^9}{\eta(3\tau)^3}$ is a weight 3 Eisenstein series for $\Gamma_1(3)$ and $\tau_o = -\frac{1}{2} + \frac{i}{2\sqrt{3}}$ is the orbifold point. The geometric rays in the scattering diagram are then the real codimension-one loci

$$\mathcal{R}_\psi(\gamma) = \{\tau : \Re(e^{\mathrm{i}\psi} Z(\gamma) = 0\} \tag{1.5}$$

For the large volume central charge and $\psi = 0$, the scattering diagram reduces to the one constructed in[4]. Finally, around the orbifold slice, we consider a two-dimensional slice $\theta_1 + \theta_2 + \theta_3 = -\sin\epsilon$ in the space of King stability parameters $\theta_i = \Im(e^{-\mathrm{i}\epsilon} Z_\tau(\mathrm{ch}\, E_i))$, and the rays are the real-codimension one loci $n_1\theta_1 + n_2\theta_2 + n_3\theta_3 = 0$.

The package file `P2Scattering.m` and various example files can be obtained from `https://github.com/bpioline/P2Scattering`

1.1 Basic usage

Assuming that the file `P2Scattering.m` is present in the user's Mathematica Application directory, the package is loaded by entering

```
In[1]:=  <<P2Scattering`
Out[1]:= P2Scattering 1.5 -- A package for evaluating DT invariants
         on K_P2
```

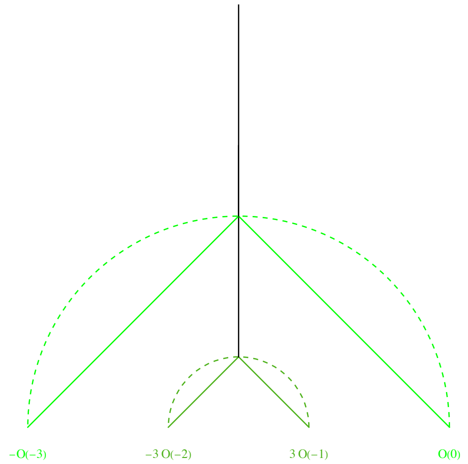
If the file `P2Scattering.m` has not yet been copied in the user's Mathematica Application directory but is in the same directory as the notebook, evaluate instead

```
In[1]:=  SetDirectory[NotebookDirectory[]]; <<P2Scattering`
Out[1]:= P2Scattering 1.2- A package for evaluating evaluating DT
         invariants on K_P2
```

For given charge $\gamma = [r, d, \chi)$ and point (s, t) on the large volume slice, the trees contributing to the index $\Omega_{(s,t)}(\gamma)$ can be found by using the routine `ScanAllTrees`, for example for $\gamma = [3, 0, 0)$ through the point $(s, t) = (-\frac{3}{2}, 2)$,

```
In[1]:=  LiTrees=ScanAllTrees[{0, 3, 0}, {-3/2, 2}]
Out[1]:= {{-3 Ch[-2], 3 Ch[-1]}, {-Ch[-3], Ch[0]}}
```

```
In[1]:=  ScattDiagLV[LiTrees, 0]
Out[1]:=
```




```
In[1]:=  Limit[EvaluateKronecker[ScattIndex[LiTrees]], y -> 1]
Out[1]:= {18,9}
```

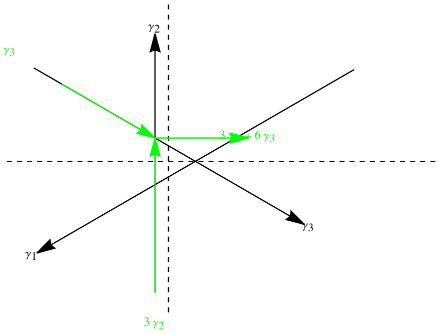
reproducing the GV invariant $N_3^0 = 27$. It should be stressed that the current implementation of the routine `ScattIndex` assumes that the index associated to each tree is a product of Kronecker indices associated to each vertex, and may give the wrong result if some of the edges carry non-primitive charges. In the case above, it does produce the correct results for both trees $K_3(3,3) + K_{12}(1,1)$. **As of version 1.5, the correct way to compute the index in presence of non-primitive intermediate states is to use the routine `ScattIndexImproved`, developed by Rishi Raj. In order to determine the trees contributing to a given charge vector at a given point (s,t), it is also more efficient to use the new routines `ConstructLVDiagram` and `LVTreesFromListRays`.**

Similarly, one can find the trees contributing near the orbifold point using `McKayScanAllTrees`: for the same charge, corresponding to dimension vector $(0, 3, 6)$, a single tree contributes in the anti-attractor chamber, with index 18,

```
In[1]:=  LiTrees = McKayScanAllTrees[chiton[{0, 3, 0}]]; LiTrees /.
         McKayrep
Out[1]:= {{3γ2, 6γ3}}
```

```
In[1]:=   Limit[EvaluateKronecker[McKayScattIndex[LiTrees]], y ->1]
Out[1]:=  {18}
```

```
In[1]:=   Show[McKayInitialRays[2], McKayScattDiag[LiTrees]]
```




```
Out[1]:=
```

As of version 1.5, the correct way to compute the index in presence of non-primitive intermediate states is to use `McKayScattIndexImproved`. In order to determine the trees contributing to a given dimension vector in the anti-attractor chamber, it is also more efficient to use the new routines `ConstructMcKayDiagram` and `McKayTreesFromListRays`.

More examples can be found on the GitHub repository.

1.2 Online documentation

The package contains many more routines, documented below, which can be used independently. Basic inline documentation can be obtained by typing e.g.

```
In[1]:=   ?EichlerT
Out[1]:=  EichlerT[tau_] gives numerical value of T[tau] using Eichler
          integral, by mapping back to fundamental domain F_C
```

1.3 History

The first version of this package was released together with the preprint [1].

2. Symbols and global variables

- `tau`: Kähler modulus in Poincaré upper half-plane
- `tau1`: Real part of `tau`
- `tau2`: Imaginay part of `tau`
- `taup`: Fricke transform $\tau' = -1/(3\tau)$

- `y`: refinement parameter
- `a`: Parameter running from 0 to 1 along rays
- `Ch[`m_`]`:denotes the charge vector for $\mathcal{O}(m)$ if m is integer; More generally, if m is rational, it denotes the charge of the image of $\mathcal{O}$ which is massless at $\tau = m$
- `Ch[`m_`][`n_`]`:denotes the charge vector for $\mathcal{O}(m)[n]$
- `Kr`$_m$`[`p_,q_`]`:denotes the index of the Kronecker quiver with m arrows, dimension vector (p, q)
- `gam1`: Charge vector $[-1, 0, -1)$ for $E_1[-1] = \mathcal{O}(0)[-1]$
- `gam2`: Charge vector $[2, -1, 0)$ for $E_2[-1] = \Omega(1)[0]$
- `gam3`: Charge vector $[-1, 1, 0)$ for $E_3[-1] = \mathcal{O}(-1)[1]$
- `tau0`: orbifold point $-1/2 + i/(2\sqrt{3})$
- `QuantumVolume`: Quantum period $\mathcal{V} = \Im T(0) \simeq 0.462758$
- `MLV`: Monodromy matrix in $[r, d, \chi)$ basis around $\tau = i\infty$
- `MCon`: Monodromy matrix in $[r, d, \chi)$ basis around at $\tau = 0$
- `MOrb`: Monodromy matrix in $[r, d, \chi)$ basis around orbifold point τ_o
- `MOrbp`: Monodromy matrix in $[r, d, \chi)$ basis around orbifold point $\tau_o + 1$
- `LVTrees[`r_,d_,chi_`]`:gives the list of precomputed trees at large volume, when available
- `McKayTrees[{`$n1$_,$n2$_,$n3$_`}]`:gives the list of precomputed trees around the orbifold point, when available
- `ExcepSlopes`: List of slopes of exceptional bundles between -3 and 4
- `ListSubsetsAny`: Precomputed list of all binary splits of Range[n] for n=2..10, used by `ListStableTrees`
- `FourierC`: List of the first 50 Fourier coefficients of Eisenstein series $C(\tau)$
- `FourierCp`: List of the first 50 Fourier coefficients of Eisenstein series $C'(\tau')$

3. Operations on Chern vectors and dimension vectors

- `Euler[{`$r\_$,$d\_$,$chi\_$`},{`$rr\_$,$dd\_$,$cchi\_$`}]`: computes the Euler form on $D^b(\mathbb{P}^2)$
- `DSZ[{`$r\_$,$d\_$,$chi\_$`},{`$rr\_$,$dd\_$,$cchi\_$`}]`: computes the antisymmetrized Euler form $3(rd' - r'd)$ on $D^b(\mathbb{P}^2)$
- `McKayDSZ[{`$n1\_$,$n2\_$,$n3\_$`},{`$nn1\_$,$nn2\_$,$nn3\_$`}]`: computes the antisymmetrized quiver Euler form
- `Disc[{`$r\_$,$d\_$,$chi\_$`}]`: computes the discriminant $\Delta(\gamma) = d^2 - 2r(\chi - r - \frac{3}{2}d))/(2r^2)$
- `Disch2[{`$r\_$,$d\_$,$ch2\_$`}]`: computes the discriminant $\Delta(\gamma) = (d^2 - 2r\,\mathrm{ch}_2)/(2r^2)$
- `DiscR[{`$r\_$,$d\_$,$chi\_$`}]`: computes the rescaled discriminant $r^2\Delta(\gamma) = d^2 - 2r(\chi - r - \frac{3}{2}d)$
- `GenSlope[{`$r\_$,$d\_$,$chi\_$`}]`: computes the slope d/r if $r \neq 0$, or $\chi/d - \frac{3}{2}$ if $r = 0$
- `DimGieseker[{`$r\_$,$d\_$,$chi\_$`}]`: computes expected dimension of moduli space of Gieseker-semistable sheaves
- `DimMcKay[{`$n1\_$,$n2\_$,$n3\_$`}]`: computes the dimension of quiver moduli space in anti-attractor chamber
- `ch2tochi[`$r\_$,$d\_$,$ch2\_$`]`: produces corresponding $[r, d, \chi)$
- `chitoch2[`$r\_$,$d\_$,$chi\_$`]`: produces corresponding $[r, d, \mathrm{ch}_2]$
- `chiton[{`$r\_$,$d\_$,$chi\_$`}]`: produces the corresponding dimension vector (n_1, n_2, n_3)
- `ntochi[{`$n1\_$,$n2\_$,$n3\_$`}]`: produces the corresponding charge vector $[r, d, chi)$
- `SpecFlow[{`$r\_$,$d\_$,$ch2\_$`},`$k\_$`]`: computes the translated charge vector $[r, d, \chi)(k)$
- `SpecFlowch2[{`$r\_$,$d\_$,$ch2\_$`},`$k\_$`]`: computes the translated charge vector $[r, d, \mathrm{ch}_2](k)$
- `repCh[]`: Replacement rule mapping `Ch`$[s]$ or `Ch`$[s][n]$ to their charge vector $[r, d, \chi)$
- `repCh2[]`: Replacement rule mapping `Ch`$[s]$ or `Ch`$[s][n]$ to their charge vector $[r, d, \mathrm{ch}_2]$
- `repChn[]`: Replacement rule mapping $Ch[m]$ into `chiton`$[\{1, m, 1+m(m+3)/2\}]$
- `repChO[]`: Replacement rule mapping $Ch[m]$ to string $O(m)$
- `repKr[]`: replaces $\mathtt{Kr}_m[p, q]$ by 1
- `LPCurve[`$mu\_$`]`: computes the Drezet-Le Potier curve $\delta(\mu)$
- `ExcepToChi[`$mu\_$`]`: gives the Chern vector $[r, d, \chi)$ of the exceptional bundle of slope μ
- `CPointchi[`$tau\_$`]`: gives the charge vector $[r, d, \chi)$ of an object that becomes massless at τ (assuming τ is a rational number)
- `CPointch2[`$tau\_$`]`: gives the charge vector $[r, d, \mathrm{ch}_2]$ of an object that becomes massless at τ (assuming τ is a rational number)
- `McKayrep[]`: replaces $\{n1\_, n2\_, n3\_\}$ by $n_1\gamma_1 + n_2\gamma_1 + n_3\gamma_3$

4. Kähler moduli space and central charge for Π-stability

- `ToFundDomainO[`*tau_*`]`:produces $\{\tau', M'\}$ such that $M' \cdot \tau' = \tau$ and τ' lies in fundamendal domain of $\Gamma_1(3)$ centered around orbifold
- `ToFundDomainC[`*tau_*`]`: produces $\{\tau', M'\}$ such that $M' \cdot \tau' = \tau$ and τ' lies in fundamendal domain of $\Gamma_1(3)$ centered around conifold
- `ToFundDomainCSeq[`*tau_*`]`:produces $\{\tau', M'\}$ such that $M' \cdot \tau' = \tau$ and τ' lies in fundamendal domain of $\Gamma_1(3)$ centered around conifold, M is a string of U,V,T[m] generators
- `ToFundDomainOApprox[`*tau_*,*precision_*`]`:produces $\{\tau', M'\}$ such that $M' \cdot \tau' = \tau$ and τ' lies in fundamendal domain of $\Gamma_1(3)$ centered around orbifold
- `ToFundDomainCApprox[`*tau_*,*precision_*`]`:produces $\{\tau', M'\}$ such that $M' \cdot \tau' = \tau$ and τ' lies in fundamendal domain of $\Gamma_1(3)$ centered around conifold
- `ApplyGamma13Lift[`*M_*,*tau_*`]`:produces the image of τ under the 3x3 monodromy matrix M
- `MonodromyOnCharge[`*M_*,{*r_*,*d_*,*chi_*}`]`:computes the image of charge vector $[r, d, \chi)$ under sequence of monodromies
- `MonodromyOnTau[`*M_*,*tau_*`]`:computes the image of τ under sequence of monodromies
- `FundDomainO[`*k_*`]`:produces the Graphics directives for the fundamental domain of $\Gamma_1(3)$ on the interval $[-\frac{1}{2}+k, \frac{1}{2}+k]$
- `FundDomainC[`*k_*`]`:produces the Graphics directives for the fundamental domain of $\Gamma_1(3)$ on the interval $[k, k+1]$
- `FundDomain3[`*k_*`]`:produces the Graphics directives for the fundamental domain of $\Gamma_1(3) + \mathbb{Z}_3$ images on the interval $[k, k+1]$
- `FundDomainRulesC[`*k_*`]`:gives a list of rules $\tau \to \tau(a)$ for boundaries of `FundDomainC[k]` parametrized by $0 < a < 1$
- `FundDomainRulesO[`*k_*`]`: gives a list of rules $\tau \to \tau(a)$ for boundaries of `FundDomainO[k]` parametrized by $0 < a < 1$
- `FundDomainRules3[`*k_*`]`:gives a list of rules $\tau \to \tau(a)$ for boundaries of `FundDomain3[k]` parametrized by $0 < a < 1$
- `EichlerC[`*tau_*`]`:numerically evaluates the Eisenstein series $C(\tau)$
- `EichlerCp[`*taup_*`]`:numerically evaluates the Eisenstein series $C'(\tau')$
- `Eichlerf1[`*tau_*`]`: evaluates $2\pi i(\tau + 1/2) + \bar{f}_1(\tau)$
- `Eichlerf2[`*tau_*`]`:evaluates $\frac{1}{2}(2\pi i)^2(\tau + \frac{1}{2})^2 + 2\pi i(\tau + \frac{1}{2})\bar{f}_1(\tau) + \bar{f}_2(\tau)$
- `Eichlerf1b[`*tau_*`]`:$= \sum_{n\geq 1} \frac{c_n}{n} e^{2\pi i n\tau}$

- `Eichlerf2b`[*tau_*]:$= -\sum_{n\geq 1} \frac{c_n}{n^2} e^{2\pi i n\tau}$
- `Eichlerf1p`[*taup_*]:$= \sum_{n\geq 1} \frac{c_n}{n} e^{2\pi i n\tau}$
- `Eichlerf2p`[*taup_*]:$= 2\pi i\tau' f_1'(\tau') - \sum_{n\geq 1} \frac{c_n}{n^2} e^{2\pi i n\tau}$
- `EichlerTLV`[*tau_*]:gives numerical value of $T(\tau)$ using Eichler integral near LV point
- `EichlerTDLV`[*tau_*]:gives numerical value of $T_D(\tau)$ using Eichler integral near LV point
- `EichlerTC`[*tau_*]:gives numerical value of $T(\tau)$ using Eichler integral near conifold point
- `EichlerTDC`[*tau_*]:gives numerical value of $T_D(\tau)$ using Eichler integral near conifold point
- `EichlerZ`[{*r_*,*d_*,*chi_*},*tau_*]:gives numerical value of $Z_\tau(\gamma)$ by mapping τ back to fundamental domain $\mathcal{F}_C$
- `EichlerZch2`[{*r_*,*d_*,*ch2_*},*tau_*]:gives numerical value of $Z_\tau(\gamma)$ by mapping τ back to fundamental domain $\mathcal{F}_C$
- `EichlerZch2LV`[{*r_*,*d_*,*ch2_*},*tau_*]:gives numerical value of $Z_\tau(\gamma)$ using Eichler integral at large volume
- `EichlerT`[*tau_*]:gives numerical value of $T(\tau)$ using Eichler integral, by mapping back to fundamental domain $\mathcal{F}_C$
- `EichlerTD`[*tau_*]:gives numerical value of $T_D(\tau)$ using Eichler integral, by mapping back to fundamental domain $\mathcal{F}_C$
- `EichlerTtilde`[*tau_*]:gives numerical value of $\tilde{T} = \frac{1}{2\sqrt{3}}(T - \frac{1}{2})$ using Eichler integral, by mapping back to fundamental domain $\mathcal{F}_C$
- `EichlerTDtilde`[*tau_*]: gives numerical value of $\tilde{T}_D = T_D - \frac{1}{2}T - \frac{1}{12}$ using Eichler integral, by mapping back to fundamental domain $\mathcal{F}_C$
- `MeijerT`[*z_*]:computes $T(z)$ using MeijerG function representation, valid for $\Im z > 0$
- `MeijerTD`[*z_*]:computes $T_D(z)$ using MeijerG function representation, valid for $\Im z > 0$
- `tauFromz`[*z_*]:computes $\tau(z)$ using ratio of hypergeometric functions, valid for $\Im z > 0$
- `zFromtau`[*z_*]:computes $z = -1/(J_3(\tau) + 15)$

5. Large volume scattering diagram

- `ZLV[{r_,d_,chi_},{s_,t_}]`: computes the large volume central charge $-\frac{1}{2}r(s+it)^2+d(s+it)-(r-\frac{3}{2}d-\chi)$
- `ZLVch2[{r_,d_,ch2_},{s_,t_}]`: computes the central charge $-\frac{1}{2}r(s+it)^2+d(s+it)-\mathrm{ch}_2$
- `Wall[{r_,d_,chi_},{rr_,dd_,cchi_},{s_,t_}]`: computes $\Im[Z(\gamma)\overline{Z(\gamma')}]$
- `Wallch2[{r_,d_,ch2_},{rr_,dd_,cch2_},{s_,t_}]`: computes $\Im[Z(\gamma)\overline{Z(\gamma')}]$
- `TreeCharge[tree_]`: computes the total charge $[r,d,\chi)$ carried by tree (or list of trees).
- `TreeChargech2[tree_]`: computes the total charge $[r,d,\mathrm{ch}_2]$ carried by tree (or list of trees).
- `TreeTop[tree_]`: computes the (s,t) coordinate of the root of the tree
- `TreeConstituents[tree_]`: gives the flattened list of constituents of Tree
- `FlipTree[tree_]`: constructs the reflected tree under $Ch[m]\to -Ch[-m]$
- `ShiftTree[tree_,k_]`: constructs the shifted tree under $Ch[m]\to Ch-m+k]$
- `ScattCheck[tree_]`: returns $\{charge,\{x,y\}\}$ of the root vertex if `Tree` is consistent, otherwise $\{totalcharge,\{\}\}$
- `ScattSort[Litree_]`: sorts trees by decreasing wall radius
- `ScattGraph[tree_]`: extracts the list of vertices and adjacency matrix of Tree
- `xytost[{x_,y_}]`: computes $\{x,\sqrt{x^2+2y}\}$
- `sttoxy[{s_,t_}]`: computes $\{s,-\frac{1}{2}(s^2-t^2)\}$
- `IntersectRays[{r_,d_,chi_},{rr_,dd_,cchi_},z_,zz_]`: returns intersection point (x,y) of two rays if the intersection point lies upward from z and z', or the empty set otherwise; the arguments z,z' can be omitted.
- `IntersectRaysSt[{r_,d_,chi_},{rr_,dd_,cchi_},psi_]`: returns intersection point (s,t) for scattering rays with phase ψ, or the empty set if they are collinear
- `TestBranch[{r_,d_,chi_},s_]`: returns `True` if $(s,\cdot)$ is on the branch with $\Im Z(\gamma)>0$, `False` otherwise
- `Rayt[{r_,d_,chi_},s_,psi_]`: computes $t(s)$ for the ray $\Re[e^{-i\psi}Z^{\mathrm{LV}}(\gamma)]=0$
- `Rays[{r_,d_,chi_},t_,psi_]`: computes $s(t)$ for the ray $\Re[e^{-i\psi}Z^{\mathrm{LV}}(\gamma)]=0$
- `Raytch2[{r_,d_,chi_},s_,psi_]`: computes $t(s)$ for the ray $\Re[e^{-i\psi}Z^{\mathrm{LV}}(\gamma)]=0$
- `Raysch2[{r_,d_,chi_},t_,psi_]`: computes $s(t)$ for the ray $\Re[e^{-i\psi}Z^{\mathrm{LV}}(\gamma)]=0$

- `ListStableTrees`[*LiCh_*,{*s0_*,*t0_*}]:constructs consistent trees from constituents in `LiCh` of the form $k_i Ch[m_i]$, which are stable at (s_0, t_0)
- `ListStableTreesPerturb`[*LiCh_*,{*s0_*,*t0_*}]:constructs consistent trees from constituents in `LiCh` of the form $k_i Ch[m_i]$, which are stable at (s_0, t_0) after perturbing ch_2 for each $Ch[m_i]$
- `ListStablePlanarTrees`[*LiCh_*,{*s0_*,*t0_*}]:constructs consistent planar trees from constituents in LiCh of the form $k_i Ch[m_i]$, which are stable at (s_0, t_0)
- `ScanConstituents`[{*r_*,*d_*,*chi_*},{*mmin_*,*mmax_*}, {*nmin_*,*nmax_*},*phimax_*]: searches possible list of constituents with slope in $[mmin, mmax]$, number of constituents in [`nmin`, `nmax`], electric potential less than `phimax` and charges adding up to $[r, d, \chi)$
- `ScanAllTrees`[{*r_*,*d_*,*chi_*},{*s_*,*t_*}]: constructs all possible trees with charges adding up to $[r, d, \chi)$ leading to an outgoing ray through the point (s, t); uses `ScanConstituents` with the most conservative values $mmin = s-t, mmax = s+t$, $nmin = 1, nmax = 2\varphi_s(\gamma)$ and $phimax = \varphi_s(\gamma) = d - rs$
- `ScanBinarySplits`[{*r_*,*d_*,*chi_*},*t0_*]:produces list of $[r', d', \chi')$ such that $[r, d, \chi)$ can split into $[r', d', \chi')$ along the ray $\mathcal{R}_0(\gamma)$ starting at height t_0
- `ScanKroneckerSplits`[{*r_*,*d_*,*chi_*}]:scans all possible pairs $\{-k'Ch(m'), kCh(m)\}$ such that each pair scatters into $[r, d, \chi)$
- `ScattDiag`[*TreeList_*]: draws scattering diagram in (x, y) plane for each tree in `TreeList`
- `ScattDiagOverlay`[*TreeList_*]: overlays scattering diagrams in (x, y) plane for each tree in `TreeList`
- `ScattDiagxy`[*TreeList_*,*psi_*]: draws scattering diagrams in (x, y) plane overlaying each tree in `TreeList`, marking the initial points
- `ScattDiagInternal`[*Tree_*]: constructs a list {total charge, coordinate of root, list of line segments, {min(x), max(x)}} for large volume scattering in (x,y) coordinates; used by `ScattDiag`
- `ScattDiagSt`[*TreeList_*,*psi_*]: draws scattering diagram in (s, t) plane for each trees in `TreeList`, approximating each edge by a straight line and overlaying the various trees; if ψ is omitted, it is assumed to vanish.
- `ScattDiagInternalSt`[*Tree_*,*psi_*]: constructs a list {total charge, coordinate of root, list of line segments, {min(s), max(s)}} for large volume scattering in (s,t) coordinates, , approximating each edge by a straight line; if ψ is omitted, it is assumed to vanish; used by `ScattDiagSt`
- `ScattDiagLV`[*TreeList_*,*psi_*]:draws scattering diagram in (s, t) in (s, t) plane for each trees in `TreeList`, using exact hyperbolaes for rays and overlaying the various trees

- `ScattDiagInternalLV`[*Tree_*,*psi_*,*Style_*]: constructs total charge, coordinate of root and list of line segments in (s,t) coordinates, $\{\min(x), \max(x)\}$, using PlotStyle→Style for plotting rays ; used by `ScattDiagLV`
- `ScattDiagLZ`[*TreeList_*]: draws scattering diagram in Li-Zhao (s,q) plane for each trees in `TreeList`
- `ScattPhi`[*TreeList_*]: overlays scattering diagrams in (s,φ) plane for each tree in `TreeList`
- `ScattPhiInternal`[*Tree_*]:constructs a list {total charge, coordinate of root, list of line segments, {min(s), max(s)}} in (s,φ) coordinates; used by `ScattPhi`
- `PlotWallRay`[{*r_*,*d_*,*chi_*},{*rr_*,*dd_*,*cchi_*}, *psi_*,{*L1_*,*L2_*,*H_*}]:plots the local scattering $[r,d,\chi) \to [r',d',\chi') + [r-r',d-d',\chi-\chi')$ for phase ψ in range $L_1 < s < L_2, 0 < t < H$
- `WallCircle`[[{*r_*,*d_*,*chi_*},{*rr_*,*dd_*,*cchi_*}]: constructs the graphics directive for the wall of marginal stability in (s,t) coordinates
- `WallLine`[[{*r_*,*d_*,*chi_*},{*rr_*,*dd_*,*cchi_*}]: constructs the graphics directive for the wall of marginal stability in (s,q) coordinates
- `TreeHue`[*i_*,*n_*]: specifies the color for the i-th tree among a list of n - can be modified at will
- `InitialLabelPosition`[*m_*]:returns a position (s,t) for the label for an initial ray with slope m; this position is lowered vertically on each call, using variables `LiSlopes` and `LiHeights` to keep track of earlier calls
- `ConstructLVDiagram`[*smin_*,*dmax_*,*phimax_*,*Nmax_*,*ListRays_*]:Constructs the large volume scattering diagram truncated up to $\varphi \le \phi_{max}$, starting from initial rays $\pm\mathcal{O}(k)$ in the range $s_{min} \le k \le s_{max}$, with scattering products $N_1\gamma_1 + N_2\gamma_2$, $2 \le N_1 + N_2 \le N_{max}$ at each vertex. The output consists of a list of $\{(r,d,\chi),\{x,y\},p_1,p_2,n_1,n_2,d\}$ where (u,v) is the initial point for the ray emitted from n_1 copies of p_1-th ray and n_2 copies of p_2-th ray at depth $d = d_1 + d_2 + 1$ (with $p_1 = p_2 = d_1 = d_2 = 0$ for initial rays); If `ListRays` is not empty, then use it as initial rays.
- `LVTreesFromListRays`[*ListRays_*,{*r_*,*d_*,*chi_*}]: extracts the list of distinct trees with given Chern character
- `ScattIndexImproved`[*TreeList_*]: computes the index for each tree in `TreeList`, taking care of non-primitive internal states
- `KroneckerTabs`[*m_*,*Nmax_*]:gives the list of populated dimension vectors $\{N_1, N_2\}$ for Kronecker quiver with m arrows, with $0 < N_1 + N_2 < N_{max}$, restricting to (N_1, N_2) coprime.
- `CostPhi`[{*r_*,*d_*,*chi_*},*s_*]$:= d - rs$

6. Orbifold scattering diagram

- `McKayRayEq`[{*n1_*,*n2_*,*n3_*},{*u_*,*v_*}]: gives the linear form vanishing on the scattering ray
- `McKayVec`[McKayVec[{*n1_*,*n2_*,*n3_*}]: computes the positive vector along the ray
- `McKayRay`[{*n1_*,*n2_*,*n3_*},{*u_*,*v_*},{vardefk1,*k2_*},*tx_*]: produces an arrow from $(u,v)+k_1w$ to $(u,v)+k_2w$, where w is the positive vector along the ray, decorated with text `tx` at the target
- `McKayListAllConsistentTrees`[{*n1_*,*n2_*,*n3_*}]: generates consistent scattering trees with leaves carrying charge $\{p,0,0\}$, $\{0,p,0\}$, $\{0,0,p\}$ adding up to (n_1,n_2,n_3), with non-zero DSZ pairing at each vertex, with distinct support
- `McKayListAllTrees`[{*n1_*,*n2_*,*n3_*}]: generates all trees with leaves carrying charge $\{p,0,0\}$, $\{0,p,0\}$, $\{0,0,p\}$ adding up to $(n_1,n_2,n_3)$and with non-zero DSZ pairing at each vertex
- `McKayScattCheck`[*Tree_*]: returns $\{charge,\{u,v\}\}$ of the root vertex if Tree is consistent, otherwise $\{totalcharge,\{\}\}$
- `McKayScattGraph`[*Tree_*]: extracts the list of vertices and adjacency matrix of Tree
- `McKayScattDiag`[*TreeList_*]: draws McKay scattering diagram in (u,v) plane for each tree in `TreeList`
- `McKayScattDiagInternal`[*Tree_*]: constructs total charge, coordinate of root and list of line segments in (u,v) coordinates; used by `McKayScattDiag`
- `McKayIntersectRays`[{*n1_*,*n2_*,*n3_*},{*nn1_*,*nn2_*,*nn3_*}]: returnsthe intersection point (u,v) of two rays, or empty set if they are collinear
- `McKayIntersectRays`[{*n1_*,*n2_*,*n3_*},{*nn1_*,*nn2_*,*nn3_*},*z_*,*zz_*]: returns intersection point (u,v) of two rays if the intersection point lies upward from z and z', or empty set otherwise
- `McKayInitialRays`[*psi_*,*L_*]: draws the initial rays in (u',v') plane, rescaling each arrow by a factor of L. If the argument ψ is omitted, it is assumed to be $\frac{\pi}{2}$.
- `ConstructMcKayDiagram`[*phimax_*,*ListRays_*]: Constructs the orbifold quiver scattering diagram with height up to `phimax`. The output consists of a list of $\{(n_1,n_2,n_3),\{u,v\},p_1,p_2,n_1,n_2\}$ where (u,v) is the initial point for the ray emitted from n_1 copies of ray p_1 and n_2 copies of ray p_2; If `ListRays` is not empty, then use it as initial rays."
- `McKayTreesFromListRays`[*ListRays_*,*Nvec_*]: extracts the list of distinct trees with given dimension vector
- `McKayScattIndexImproved`[*TreeList_*]: computes the index for each tree in `TreeList`, taking care of non-primitive internal states
- `InitialRaysOrigin`: is a List of initial points (u_i,v_i) for the 3 initial rays in the orbifold scattering diagram

7. Exact scattering diagram

- `CriticalPsi`[*mu_*]$:=\arctan(\mu/\mathcal{V})$
- `IntersectExactRaysLV`[{*r_*,*d_*,*chi_*},{*rr_*,*dd_*,*cchi_*},*psi_*]:returns value of τ at intersection point of two exact rays using `EichlerTLV` to evaluate the periods, or 0 if they are collinear
- `IntersectExactRaysC`[{*r_*,*d_*,*chi_*},{*rr_*,*dd_*,*cchi_*},*psi_*]: returns value of τ atf intersection point of two exact rays using `EichlerTC` to evaluate the periods, or 0 if they are collinear
- `XY`[*tau_*,*psi_*]:computes the affine coordinates (x,y) such that scattering rays are straight lines $ry+dx-\mathrm{ch}_2=0$
- `CPointxy`[*tau_*]:computes the (x,y) coordinate of initial point $Ch[\tau]$ (assuming τ is a rational number)
- `IntegralCurve`[*tauinit_*,*tangent_*,{*ainit_*,*amin_*,*amax_*},*boundaries_*]:produces a function $f:a\in[0,1]\to\mathbb{H}$ with $f(ainit)=tauinit$ following the tangent direction (an expression in τ) and stopping at the boundaries (by default: $\{Im\tau=0.01\}$). The range of integration parameters $\{amin,amax\}$ can be infinite provided the actual rays remain finite by hitting the boundaries.
- `NormalizeFunctionDomain`[*fun_*]:rescales the argument of the `InterpolatingFunction` `fun` to interval $[0,1]$
- `DtauT`[*tau_*]:numerically evaluates $\partial_\tau T(\tau)$
- `DtauZch2`[{*r_*,*d_*,*ch2_*}m*tau_*]:numerically evaluates $\partial_\tau Z_\tau(\gamma)$
- `DtauZ`[{*r_*,*d_*,*chi_*},*tau_*]:numerically evaluates $\partial_\tau Z_\tau(\gamma)$
- `ArgDtauT`[vardeftau]:computes the argument of $T'(\tau)$, between $-\pi$ and π
- `ArgDtauTD`[vardeftau]:computes the argument of $T'_D(\tau)$, between $-\pi$ and π
- `ArgDtauZch2`[{*r_*,*d_*,*ch2_*},*tau_*]:computes the argument of $\partial_\tau Z_\tau(\gamma)$, between $-\pi$ and π
- `UnitDtauT`[vardeftau]:numerically evaluates $\partial_\tau T(\tau)/|\partial_\tau T(\tau)|$
- `UnitDtauDT`[vardeftau]:numerically evaluates $\partial_\tau T_D(\tau)/|\partial_\tau T_D(\tau)|$
- `UnitDtauZ`[{*r_*,*d_*,*chi_*}*tau_*]:numerically evaluates $\partial_\tau Z_\tau(\gamma)/|\partial_\tau Z_\tau(\gamma)|$
- `UnitDtauZch2`[{*r_*,*d_*,*ch2_*}*tau_*]:numerically evaluates $\partial_\tau Z_\tau(\gamma)/|\partial_\tau Z_\tau(\gamma)|$
- `NormalizeApprox`[*z_*,*eps_*]:normalizes $z\in\mathbb{C}$ to roughly unit length for large z, but behaves smoothly near zero.
- `TotalChargech2`[*Tree_*]: gives the total charge vector $[r,d,\mathrm{ch}_2]$ of a given tree (nested list).

- `TotalChargechi`[*Tree_*]: gives the total charge vector $[r, d, \chi)$ of a given tree (nested list).

- `ConifoldRay`[*init_*,*psi_*,*homshift_*]: gives a function $f : a \in [0,1] \to \mathbb{H}$ parametrizing the ray starting at the rational number init$= \frac{p}{q}$ (with $q \neq 0 \mod 3$

- `RayCh`[*psi_*]: gives an initial ray starting at 0, namely a function $f : [0,1] \to \mathbb{H}$ starting (close to) 0 and following a ray where $Z_\tau([1,0,0]) = -T_D$ has phase $\psi + \frac{\pi}{2} \mod \pi$. Shifting ψ by 2π gives a different ray, corresponding to a homological shift by 2. Values are cached.

- `RayGeneralch2`[*psi_*,*tauexpr_*,*start_*]: gives a function $a \in [0,1] \to \mathbb{H}$ parametrizing the ray where $Z_\tau(\gamma)$ has phase $\psi + \frac{\pi}{2} \mod \pi$. The starting point is obtained using `FindRoot`$[\dots, Z_{tauexpr}(\gamma), \dots, start]$, see documentation of `FindRoot`.

- `RayFromInfinity`[{*r_*,*d_*,*chi_*},*psi_*]: gives a function $f : a \in [0,1] \to \mathbb{H}$ parametrizing the ray of phase ψ starting from the large volume limit

- `StabilityWall`[*Tree_*,*tauinit_*,*tangent_*,{*ainit_*,*amin_*,*amax_*}]: gives a a function $f : a \in [0,1] \to \mathbb{H}$ parametrizing the stability wall for the last fusion of the tree. The tree can also be a pair of charges. The tauinit is used as a starting point of `FindRoot` along a vertical line. The last argument can be omitted and defaults to $\{-2, 2\}$; it is an interval around the starting point 0, and can be used to restrict the stability wall to only one side of tauinit.

- `TreeToRays`[*Tree_*,*psi_*]: gives the (flat) list of rays (functions $f : [0,1] \to \mathbb{H}$ where $Z_\tau(\gamma_e)$ has phase $\psi + \frac{\pi}{2} \mod \pi$ along each edge with charge γ_e. The tree is given as a nested list of initial objects of the form $kCh[p/q][n]$ with k, p, q, n integers.

- `TreeToRaysPlot`[*Tree_*,*psi_*,*plotoptions_*]: Plots the rays produced by `TreeToRays[Tree,psi]` with the given plot options.

8. Index computations

- `ScattIndex`[*TreeList_*]: computes the index for each tree in `TreeList`; do not trust the result if internal lines have non-primitive charges !

- `ScattIndexInternal`[*TreeList_*]: computes total charge, list of Kronecker indices associated to each vertex in `Tree`

- `McKayScattIndex`[*TreeList_*]: computes the index for each tree in TreeList; do not trust the result if internal lines have non-primitive charges !

- `McKayScattIndexInternal`[*Tree_*]: computes total charge, list of Kronecker indices associated to each vertex in Tree

- `EvaluateKronecker[`$f\_$`]`: replaces each $Kr_m[p,q]$ with the index of the Kronecker quiver with m arrows and dimension vector (p,q), using routines taken from `CoulombHiggs.m` package
- `BinaryTreeIndex[`*TreeList_*`]`: computes the index of each binary tree in `TreeList`
- `BinaryTreeIndexInternal[`*Tree_*`]`: produces a list of wall-crossing factors from each vertex in a binary tree
- `AbelianizedScatteringSequence[`*ScattSeq_*`]`: produces a list of abelianized constituents, by replacing each $kCh[m]$ by $\{k_i Ch[m]\}$ where k_i runs over integer partitions of k
- `IndexFromAbelianizedSequences[`*LiAbelianized_*,{*s0_*,*t0_* }`]`: constructs all possible perturbed binary trees from constituents in `LiAbelianized` which are stable for the large volume central charge at (s_0,t_0), and compute the contribution to the index from each of them
- `IndexFromSequences[`*LiScattSeq_*,,{*s0_*,*t0_* }`]`: constructs all possible lists of abelianized constituents from the list of scattering sequences `LiScattSeq`, removes duplicates, and applies `IndexFromAbelianizedSequences` on each of them

9. Higgs branch formula

In addition, the package includes routines adapted from the Mathematica package `CoulombHiggs.m` [5], mainly for the purpose of evaluating the indices $K_m(p,q)$ of the Kronecker quiver. The names of the routines are prefaced by `P2` to avoid conflicts.

- `P2HiggsBranchFormula[`*Mat_*,*Cvec_*,*Nvec_* `]`: computes the refined index of a quiver with DSZ products α_{ij} = `Mat`$[[i,j]]$, dimension vector N_i = `Nvec`$[[i]]$, FI parameters ζ_i = `Cvec`$[[i]]$, using Reineke's formula. It is assumed, but not checked, that the quiver has no oriented loop;
- `P2RationalInvariant[`*Mat_*,*Cvec_*,*Nvec_*,*y_* `]`: computes the refined rational index of a quiver with DSZ matrix α_{ij} = `Mat`$[[i,j]]$, FI parameters ζ_i = `Cvec`$[[i]]$, dimension vector N_i = `Nvec`$[[i]]$, using Reineke's formula
- `P2StackInvariant[`*Mat_*,*Cvec_*,*Nvec_*,*y_* `]`: computes the stacky invariant of a quiver with DSZ matrix α_{ij} = `Mat`$[[i,j]]$, FI parameters ζ_i = `Cvec`$[[i]]$, dimension vector N_i = `Nvec`$[[i]]$, using Reineke's formula
- `P2BinarySplits[`*Nvec_*`]`: gives the list of dimension vectors γ_L less than γ, quotiented by the equivalence relation $\gamma_L \to \gamma - \gamma_L$.
- `P2ListAllPartitions[`*gam_*`]`: returns the list of unordered partitions of the positive integer vector γ as a sum of positive, non-zero integer vectors α_i;
- `P2QDeformedFactorial[`*n_*,*y_*`]`: computes the q-deformed factorial $[n,y]!$

References

[1] P. Bousseau, P. Descombes, B. Le Floch, and B. Pioline, "BPS Dendroscopy on Local P^2," 2210.10712.

[2] B. Le Floch, B. Pioline, and R. Raj, "BPS Dendroscopy on Local $\mathbb{P}^1 \times \mathbb{P}^1$," 2412.07680.

[3] P. Bousseau, P. Descombes, B. Le Floch, and B. Pioline, "BPS Dendroscopy on local $\mathbb{P}_2$." To appear.

[4] P. Bousseau, "Scattering diagrams, stability conditions, and coherent sheaves on $\mathbb{P}^2$," *J. Algebraic Geom.* **31** (2022) 593–686, 1909.02985.

[5] "`CoulombHiggs.m`, a Mathematica package for computing quiver invariants." available from `https://github.com/bpioline/CoulombHiggs`.