

LISSAFIRE: Lissajous-figure Reconstruction for nonlinear polarization tomography of bichromatic fields

© Emilio Pisanty 2019. Licensed under GPL and CC-BY-SA.

■ Introduction

Readme

LISSAFIRE is a Mathematica package for reconstructing the polarization Lissajous figures of bichromatic $\omega:2\omega$ light-beam combinations, as described in the paper

- Knotting fractional-order knots with the polarization state of light. E. Pisanty et al. *Nature Photonics*, in press (2019), arXiv:1808.05193.

```
(* LISSAFIRE: Lissajous-  
figure Reconstruction for nonlinear polarization tomography of bichromatic fields. *)  
(* © Emilio Pisanty, 2019 *)  
  
(* For more information, see https://github.com/episanty/LISSAFIRE *)
```

Licensing

This code is dual-licensed under the GPL and CC-BY-SA licenses; you are free to use, modify, and redistribute it, but you must abide by the terms in either of those licenses.

In addition to that *legal* obligation, if you use this code in calculations for an academic publication, you have an *academic* obligation to cite it correctly. For that purpose, please cite the *Nature Photonics* paper above, or use a direct citation to the code such as

E. Pisanty. LISSAFIRE: Lissajous-figure reconstruction for nonlinear polarization tomography of bichromatic fields. <https://github.com/episanty/LISSAFIRE> (2019).

If you wish to include a DOI in your citation, please use one of the numbered-version releases.

■ Implementation

Initialization and package infrastructure

Package initialization

```
BeginPackage["LISSAFIRE`"];
```

Version number

The variable `$LISSAFIREversion` gives the version of the LISSAFIRE package currently loaded, and its timestamp

```
$LISSAFIREversion::usage = "$LISSAFIREversion prints the
    current version of the LISSAFIRE package in use and its timestamp.";
$LISSAFIREtimestamp::usage = "$LISSAFIREtimestamp prints the timestamp
    of the current version of the LISSAFIRE package.";
Begin["`Private`"];
$LISSAFIREversion := "LISSAFIRE v1.0.1, "<>$LISSAFIREtimestamp;
End[];
```

The timestamp is updated every time the notebook is saved via an appropriate notebook option, which is set by the code below.

```
In[ ]:= SetOptions[
  EvaluationNotebook[],
  NotebookEventActions → {{"MenuCommand", "Save"} ⇒ (
    NotebookWrite[
      Cells[CellTags → "version-timestamp"][[1]],
      Cell[
        BoxData[RowBox[
          {"Begin[\"`Private`\"]; $LISSAFIREtimestamp = \"\" <> DateString[] <> \"\"; End[];"}]]
        , "Input", InitializationCell → True, CellTags → "version-timestamp"
      ], None, AutoScroll → False];
    NotebookSave[]
  ), PassEventsDown → True}
];
```

To reset this behaviour to normal, evaluate the cell below

```
SetOptions[EvaluationNotebook[],
  NotebookEventActions → {{"MenuCommand", "Save"} ⇒ (NotebookSave[]), PassEventsDown → True}]
```

Timestamp

```
Begin["`Private`"]; $LISSAFIREtimestamp = "Tue 23 Apr 2019 20:32:28"; End[];
```

Directory

```
$LISSAFIREdirectory::usage = "$LISSAFIREdirectory is the
    directory where the current LISSAFIRE package instance is located.";
```

```
Begin["`Private`"];
With[{softLinkTestString = StringSplit[StringJoin[ReadList[
    "! ls -la " <> StringReplace[$InputFileName, {" " → "\\ "}], String]], " -> "]},
If[Length[softLinkTestString] > 1, (*Testing in case $InputFileName
    is a soft link to the actual directory.*)
    $LISSAFIREdirectory = StringReplace[DirectoryName[softLinkTestString[[2]],
        {" " → "\\ "}],
    $LISSAFIREdirectory = StringReplace[DirectoryName[$InputFileName], {" " → "\\ "}],
    ];
End[];
```

Git commit hash and message

```
$LISSAFIREcommit::usage =
    "$LISSAFIREcommit returns the git commit log at the location of the
    LISSAFIRE package if there is one.";
$LISSAFIREcommit::OS = "$LISSAFIREcommit has only been tested on Linux.";
```

```
Begin["`Private`"];
$LISSAFIREcommit := (If[$OperatingSystem ≠ "Unix", Message[$LISSAFIREcommit::OS];
    StringJoin[
        Riffle[ReadList["!cd " <> $LISSAFIREdirectory <> " && git log -1", String], {"\n"}]]];
End[];
```

UnitE

```
UnitE::usage = "UnitE[1] and UnitE[-1] return, respectively,  $\frac{1}{\sqrt{2}}\{1, i\}$  and  $\frac{1}{\sqrt{2}}\{1, -i\}$ .";
Begin["`Private`"];
UnitE[1] =  $\frac{1}{\sqrt{2}}\{1, i\}$ ;
UnitE[-1] =  $\frac{1}{\sqrt{2}}\{1, -i\}$ ;
End[];
```

EnsureRightCircularFundamental

```
EnsureRightCircularFundamental::usage =
  "EnsureRightCircularFundamental[{E1p,E1m,E2p,E2m}] ensures that
    the fundamental has right-circular polarization (i.e. |E1p|>|E1m|)
    by swapping the input to {E1m*,E1p*,E2m*,E2p*} if necessary.";

Begin["`Private`"];
EnsureRightCircularFundamental[{E1p_, E1m_, E2p_, E2m_}] := If[
  Abs[E1p] > Abs[E1m],
  {E1p, E1m, E2p, E2m},
  {E1m*, E1p*, E2m*, E2p*}
]
End[];
```

PhaseNormalization

```
PhaseNormalization::usage =
  "PhaseNormalization[{E1p,E1m,E2p,E2m}] Normalizes the field phases
    so that E1p is real and positive, by multiplying by
    an appropriate factor of  $e^{-i\phi}$  on E1 and  $e^{-2i\phi}$  on E2.";

Begin["`Private`"];
PhaseNormalization[{E1p_, E1m_, E2p_, E2m_}] := With[{ $\phi$  = Arg[E1p]},
  Chop[Times[
    { $e^{-i\phi}$ ,  $e^{-i\phi}$ ,  $e^{-2i\phi}$ ,  $e^{-2i\phi}$ },
    {E1p, E1m, E2p, E2m}
  ]]
]
End[];
```

NLPTOutcomes

```

NLPTOutcomes::usage =
  "NLPTOutcomes[{ReE1p,ImE1p,ReE1m,ImE1m,ReE2p,ImE2p,ReE2m,ImE2m}] Calculates
    the Nonlinear Polarization Tomography outcome functions  $I_n$ , as
    defined in the paper, with the normalization set so that the
     $\ell > 0$  components of the paper are given by  $I_\ell^{\text{paper}}(\theta) = \text{Re}(2I_\ell e^{i\ell\theta})$ .

NLPTOutcomes[{E1p,E1m,E2p,E2m}] Uses explicit complex amplitudes.";

Begin["`Private`"];

NLPTOutcomes[{ReE1p_, ImE1p_, ReE1m_, ImE1m_, ReE2p_, ImE2p_, ReE2m_, ImE2m_}] =
  Block[{u, E1, E2, E1p, E1m, E2p, E2m},
    u = {Cos[θ], Sin[θ]};
    E1 = UnitE[1] E1p + UnitE[-1] E1m;
    E2 = UnitE[1] E2p + UnitE[-1] E2m;

    Table[
      Expand[
        Coefficient[
          TrigToExp[
             $\frac{1}{2} \left( (u \cdot E1^*)^2 + u \cdot E2^* \right) \left( (u \cdot E1)^2 + u \cdot E2 \right)$ 
          ] /. {θ →  $\frac{1}{i} \text{Log}[e^{i\theta}]$ }
          , e^{iθ}, n] /. {
            E1p → ReE1p + i ImE1p,
            E1m → ReE1m + i ImE1m,
            E2p → ReE2p + i ImE2p,
            E2m → ReE2m + i ImE2m
          } /. {
            Conjugate[symbol_?AtomQ] → symbol
          }
        ]
      , {n, 0, 4}
    ]
  ];

NLPTOutcomes[{E1p_, E1m_, E2p_, E2m_}] :=
  NLPTOutcomes[{Re[E1p], Im[E1p], Re[E1m], Im[E1m], Re[E2p], Im[E2p], Re[E2m], Im[E2m]}]

End[];

```

ReconstructionMimimizationTarget

```

ReconstructionMimimizationTarget::usage =
  "ReconstructionMimimizationTarget[{I0,I1,I2,I3,I4}][ReE1p,ImE1p,ReE1m,ImE1m,ReE2p,ImE2p,
  ,ReE2m,ImE2m] calculates the reconstruction target

$$\sum_{n=0}^4 |I_n - I_n(E)|^2$$
, i.e. the sum of squares of the nonlinear-polarimetry
  Fourier coefficients in difference between the given  $I_n$ 
  and those reconstructed from the given complex fields.";

Begin["`Private`"];

ReconstructionMimimizationTarget[{I0_, I1_, I2_, I3_, I4_}][ReE1p_, ImE1p_, ReE1m_,
  ImE1m_, ReE2p_, ImE2p_, ReE2m_, ImE2m_] = Block[{u, E1, E2, E1p, E1m, E2p, E2m},
  u = {Cos[θ], Sin[θ]};
  E1 = UnitE[1] E1p + UnitE[-1] E1m;
  E2 = UnitE[1] E2p + UnitE[-1] E2m;

  Total[Flatten[
    Table[
      (
        Simplify[
          ReIm[
            NLPTOutcomes[{ReE1p, ImE1p, ReE1m, ImE1m, ReE2p, ImE2p, ReE2m, ImE2m}][[n + 1]] -
              {I0, I1, I2, I3, I4}][[n + 1]]
          ]
        , Assumptions →
          {{ReE1p, ImE1p, ReE1m, ImE1m, ReE2p, ImE2p, ReE2m, ImE2m} ∈ Reals, I0 ∈ Reals}
        ]
      )^2
      , {n, 0, 4}
    ]]]
  ];

End[];

```

ReconstructBicircularField

```

Options[ReconstructBicircularFieldList] = Join[{SortingFunction → Function[#["Residual"]],
  SelectionFunction → Function[True], Parallelize → False}, Options[FindMinimum]];
Options[ReconstructBicircularField] = Options[ReconstructBicircularFieldList];

SortingFunction::usage = "SortingFunction is an option
  for ReconstructBicircularField and ReconstructBicircularFieldList

```

that specifies a function (applied to the results of the form
 $\langle | \text{"Fields"} \rightarrow \{E_{1+}, E_{1-}, E_{2+}, E_{2-}\}, \text{"Outcomes"} \rightarrow \{I_{0, \text{rec}}, I_{1, \text{rec}}, I_{2, \text{rec}}, I_{3, \text{rec}}, I_{4, \text{rec}}\}, \sqrt{\text{Residual}} \rightarrow r, \text{"Residual"} \rightarrow r^2 | \rangle$) used to sort the individual minima.";

SelectionFunction::usage =

"SelectionFunction is an option for ReconstructBicircularField and
 ReconstructBicircularFieldList that specifies

a function (applied to the results of the form

$\langle | \text{"Fields"} \rightarrow \{E_{1+}, E_{1-}, E_{2+}, E_{2-}\}, \text{"Outcomes"} \rightarrow \{I_{0, \text{rec}}, I_{1, \text{rec}}, I_{2, \text{rec}}, I_{3, \text{rec}}, I_{4, \text{rec}}\}, \sqrt{\text{Residual}} \rightarrow r, \text{"Residual"} \rightarrow r^2 | \rangle$, and returning True

or False) used to keep or discard potential solutions.";

Protect[SortingFunction, SelectionFunction];

ReconstructBicircularFieldList::usage =

"ReconstructBicircularFieldList[{I0,I1,I2,I3,I4}] calculates a list
 of candidate reconstructed fields (each an Association of the form

$\langle | \text{"Fields"} \rightarrow \{E_{1+}, E_{1-}, E_{2+}, E_{2-}\}, \text{"Outcomes"} \rightarrow \{I_{0, \text{rec}}, I_{1, \text{rec}}, I_{2, \text{rec}}, I_{3, \text{rec}}, I_{4, \text{rec}}\}, \sqrt{\text{Residual}} \rightarrow r, \text{"Residual"} \rightarrow r^2 | \rangle$), obtained by

minimizing ReconstructionMinimizationTarget over a list
 of random initial seeds pulled from a box of side 1.

ReconstructBicircularFieldList[{I0,I1,I2,I3,I4},Erangle] uses a box of side Erangle (which
 can be a single number, or a list of eight real numbers to be used as the
 sizes of the boxes for {ReE1p,ImE1p,ReE1m,ImE1m,ReE2p,ImE2p,ReE2m,ImE2m})
 for the initial seeds of the minimization.

ReconstructBicircularFieldList[{I0,I1,I2,I3,I4},Erangle,iterations]
 uses the specified number of iterations.";

ReconstructBicircularField::usage =

"ReconstructBicircularField[{I0,I1,I2,I3,I4}] returns the first
 element of the corresponding ReconstructBicircularFieldList,
 using the specified (or default) SortingFunction.

ReconstructBicircularField[{I0,I1,I2,I3,I4},Erangle] returns the
 first element of the corresponding ReconstructBicircularFieldList,
 using the specified (or default) SortingFunction.

ReconstructBicircularField[{I0,I1,I2,I3,I4},Erangle,iterations] returns the
 first element of the corresponding ReconstructBicircularFieldList,
 using the specified (or default) SortingFunction.";

Begin["`Private`"];

ReconstructBicircularField[{I0_, I1_, I2_, I3_, I4_},
 Erangle_: 1, iterations_: 20, options: OptionsPattern[]] := First[
 ReconstructBicircularFieldList[{I0, I1, I2, I3, I4}, Erangle, iterations, options]

```

]

ReconstructBicircularFieldList[{I0_, I1_, I2_, I3_, I4_},
  Erange_: 1, iterations_: 20, options: OptionsPattern[]] := Block[{ },

  SortBy[
    Select[
      If[OptionValue[Parallelize] == True, Parallelize, # &]@Table[
        Function[solution, Block[{fields},
          fields = PhaseNormalization[
            EnsureRightCircularFundamental[
              {ReE1p +  $\mathbf{i}$  ImE1p, ReE1m +  $\mathbf{i}$  ImE1m, ReE2p +  $\mathbf{i}$  ImE2p, ReE2m +  $\mathbf{i}$  ImE2m} /. solution[[2]]
            ]];
          <|"Fields" → fields, "Outcomes" → NLPTOutcomes[fields],
            " $\sqrt{\text{Residual}}$ " →  $\sqrt{\text{solution}[[1]}$ , "Residual" → solution[[1]]|>
        ]
      ]
    ],
    FindMinimum[
      ReconstructionMimimizationTarget[{I0, I1, I2, I3, I4}][
        ReE1p, ImE1p, ReE1m, ImE1m, ReE2p, ImE2p, ReE2m, ImE2m],
      Transpose[{
        {ReE1p, ImE1p, ReE1m, ImE1m, ReE2p, ImE2p, ReE2m, ImE2m},
        Erange RandomReal[{-1, 1}, 8]
      }],
      Method → {"Newton", StepControl → "TrustRegion"},
      Evaluate[Sequence @@ FilterRules[{options}, Options[FindMinimum]]]
    ]
  ], {iterations}]
  , OptionValue[SelectionFunction]]
  , OptionValue[SortingFunction]]
]

End[];

```

Package closure

End of package

```
EndPackage[];
```

Add to distributed contexts.

```
DistributeDefinitions["LISSAFIRE`"];
```