

# H0Approximator

**A package to estimate global sections of a  
pullback line bundle on hypersurface  
curves in  $\mathbb{P}^3$**

2020.06.15

15 June 2020

**Martin Bies**

**Martin Bies**

Email: [martin.bies@alumni.uni-heidelberg.de](mailto:martin.bies@alumni.uni-heidelberg.de)

Homepage: <https://martinbies.github.io/>

Address: Mathematical Institute

University of Oxford

Andrew Wiles Building

Radcliffe Observatory Quarter

Woodstock Road

Oxford OX2 6GG

United Kingdom

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	What is the goal of the H0Approximator package? . . . . .	3
<b>2</b>	<b>Spectrum approximation from curve splittings</b>	<b>4</b>
2.1	Compute if a curve of given class is irreducible . . . . .	4
2.2	Finding the CounterDirectory . . . . .	4
2.3	Determine descendant level . . . . .	4
2.4	Approximate h0-spectrum . . . . .	5
2.5	Examples . . . . .	5
<b>3</b>	<b>Spectrum approximation from maximal curve splittings</b>	<b>7</b>
3.1	Install elementary topological functions on dP3 . . . . .	7
3.2	Check if a curve class if a power of a rigid divisor . . . . .	7
3.3	Topological section counter . . . . .	8
3.4	Local section analyser . . . . .	8
3.5	Analyse bundle on maximally degenerate curves . . . . .	9
3.6	Examples . . . . .	10
	<b>Index</b>	<b>11</b>

# Chapter 1

## Introduction

### 1.1 What is the goal of the H0Approximator package?

*H0Approximator* provides functionality to estimate global sections from topological counts only. A refined approximation checks irreducibility of curves, and thereby computes more accurate results, at the expense of longer runtimes.

The current implementations is specific to hypersurface curves in dP3 and pullback line bundles thereon. Generalizations thereof are reserved for future work.

## Chapter 2

# Spectrum approximation from curve splittings

### 2.1 Compute if a curve of given class is irreducible

#### 2.1.1 IsIrreducible (for IsList, IsToricVariety)

▷ `IsIrreducible(List)` (operation)

**Returns:** True or false

This operation identifies if a curve class defines an irreducible curve or not.

#### 2.1.2 DegreesOfComponents (for IsList, IsToricVariety)

▷ `DegreesOfComponents(List)` (operation)

**Returns:** A list of fail

This operation performs a primary decomposition of a hypersurface curve in dP3. If all components are principal, it returns the degrees of the generators. Otherwise it returns fail.

### 2.2 Finding the CounterDirectory

#### 2.2.1 FindCounterBinary

▷ `FindCounterBinary(none)` (operation)

**Returns:** the corresponding filename

This operation identifies the location of the counter binary.

### 2.3 Determine descendant level

#### 2.3.1 DescendantLevel (for IsList)

▷ `DescendantLevel(List, c)` (operation)

**Returns:** An integer

Estimates the maximal power to which a rigid divisor can be peeled-off the given curve.

## 2.4 Approximate h0-spectrum

### 2.4.1 RoughApproximationWithSetups (for IsList, IsList)

▷ `RoughApproximationWithSetups(Lists, c, l)` (operation)

**Returns:** A list

Given a curve class `c` and a line bundle class `l`, this method approximates the  $h_0$ -spectrum by use of topological methods only. In particular, irreducibility of curves is not checked. Consequently, this method performs faster than `FineApproximation`, but produces less accurate results.

### 2.4.2 RoughApproximation (for IsList, IsList)

▷ `RoughApproximation(Lists, c, l)` (operation)

**Returns:** A list

The same as `RoughApproximationWithSetups`, but returns only the spectrum estimate.

### 2.4.3 FineApproximationWithSetups (for IsList, IsList)

▷ `FineApproximationWithSetups(Lists, c, l)` (operation)

**Returns:** A list

Given a curve class `c` and a line bundle class `l`, this method approximates the  $h_0$ -spectrum by use of topological methods and checks irreducibility of curves. It performs slower than `RoughApproximation`, but produces more accurate results.

### 2.4.4 FineApproximation (for IsList, IsList)

▷ `FineApproximation(Lists, c, l)` (operation)

**Returns:** A list

The same as `FineApproximationWithSetups`, but returns only the spectrum estimate.

## 2.5 Examples

We can approximate the spectrum roughly, that is we do not take irreducibility of curves into account. Here is a simple example:

Example

```
gap> approx1 := RoughApproximation( [3,-1,-1,-1],[1,-1,-3,-1] );;
(*) Curve: [ 3, -1, -1, -1 ]
(*) Bundle: [ 1, -1, -3, -1 ]
(*) 79 rough approximations
(*) Rough spectrum estimate: [ 0, 1, 2, 3 ]
    (x) h0 = 0: 22
    (x) h0 = 1: 6
    (x) h0 = 2: 37
    (x) h0 = 3: 14
```

We can of course compute this also finer, i.e. by checking irreducibility for each identified setup:

Example

```
gap> approx2 := FineApproximation( [3,-1,-1,-1],[1,-1,-3,-1] );;
(*) Curve: [ 3, -1, -1, -1 ]
```

```
(*) Bundle: [ 1, -1, -3, -1 ]
(*) 79 rough approximations
(*) Rough spectrum estimate: [ 0, 1, 2, 3 ]
  (x) h0 = 0: 22
  (x) h0 = 1: 6
  (x) h0 = 2: 37
  (x) h0 = 3: 14
(*) Checking irreducibility of curves...
(*) 23 fine approximations
(*) Fine spectrum estimate: [ 0, 2, 3 ]
  (x) h0 = 0: 11
  (x) h0 = 2: 11
  (x) h0 = 3: 1
```

Here is a more involved example:

Example

```
gap> approx2 := RoughApproximation( [5,-1,-1,-2],[1,1,-4,1] );;
(*) Curve: [ 5, -1, -1, -2 ]
(*) Bundle: [ 1, 1, -4, 1 ]
(*) 325 rough approximations
(*) Rough spectrum estimate: [ 0, 1, 2, 3, 4, 5, 6, 7 ]
  (x) h0 = 0: 13
  (x) h0 = 1: 18
  (x) h0 = 2: 9
  (x) h0 = 3: 37
  (x) h0 = 4: 30
  (x) h0 = 5: 31
  (x) h0 = 6: 148
  (x) h0 = 7: 39
```

## Chapter 3

# Spectrum approximation from maximal curve splittings

### 3.1 Install elementary topological functions on dP3

#### 3.1.1 IntersectionNumber (for IsList, IsList)

▷ `IntersectionNumber(Lists, d1, d2)` (operation)  
**Returns:** Integer  
Compute the topological intersection number between two divisor classes d1, d2 in dP3

#### 3.1.2 Genus (for IsList)

▷ `Genus(List, c)` (operation)  
**Returns:** Integer  
Compute the genus of a curve of class c in dP3

#### 3.1.3 LineBundleDegree (for IsList, IsList)

▷ `LineBundleDegree(Lists, l, c)` (operation)  
**Returns:** Integer  
Computes the degree of a pullback line bundle of class l on a curve of class c in dP3

### 3.2 Check if a curve class is a power of a rigid divisor

#### 3.2.1 IsE1Power (for IsList)

▷ `IsE1Power(List, c)` (operation)  
**Returns:** True or false  
Checks if a curve class is a power of E1

#### 3.2.2 IsE2Power (for IsList)

▷ `IsE2Power(List, c)` (operation)  
**Returns:** True or false

Checks if a curve class if a power of E1

### 3.2.3 IsE3Power (for IsList)

- ▷ `IsE3Power(List, c)` (operation)  
**Returns:** True or false  
 Checks if a curve class if a power of E1

### 3.2.4 IsE4Power (for IsList)

- ▷ `IsE4Power(List, c)` (operation)  
**Returns:** True or false  
 Checks if a curve class if a power of E1

### 3.2.5 IsE5Power (for IsList)

- ▷ `IsE5Power(List, c)` (operation)  
**Returns:** True or false  
 Checks if a curve class if a power of E1

### 3.2.6 IsE6Power (for IsList)

- ▷ `IsE6Power(List, c)` (operation)  
**Returns:** True or false  
 Checks if a curve class if a power of E1

### 3.2.7 IsRigidPower (for IsList)

- ▷ `IsRigidPower(List, c)` (operation)  
**Returns:** True or false  
 Checks if a curve class if a power of a rigid divisor.

## 3.3 Topological section counter

### 3.3.1 Sections (for IsInt, IsInt)

- ▷ `Sections(Integers, d, g)` (operation)  
**Returns:** Integer  
 Based on degree d of a line bundle and the genus g of the curve, this method tries to identify the number of sections of the line bundle.

## 3.4 Local section analyser

### 3.4.1 IntersectionMatrix (for IsList)

- ▷ `IntersectionMatrix(List, of, curve, components)` (operation)  
**Returns:** A list of lists of integers  
 Identify the intersection matrix among all components of a curve.



### 3.4.2 IntersectionsAmongCurveComponents (for IsList)

▷ `IntersectionsAmongCurveComponents(List, of, curve, components.)` (operation)

**Returns:** A list of integers

Identify the intersection numbers among all components of a curve.

### 3.4.3 EstimateGlobalSections (for IsList, IsList)

▷ `EstimateGlobalSections(Lists, L1, L2)` (operation)

**Returns:** An integer

This method estimates the number of global sections based on the list L1 of local sections and the list L2 of intersection numbers among the split components of the curve.

### 3.4.4 IsSimpleSetup (for IsList, IsList)

▷ `IsSimpleSetup(Lists, S, n)` (operation)

**Returns:** An integer

This method checks whether the pair of curve with components with intersection numbers I and local section counts n allow to easily estimate the number of global sections.

### 3.4.5 AnalyzeBundleOnCurve (for IsList, IsList)

▷ `AnalyzeBundleOnCurve(Lists, S, l)` (operation)

**Returns:** An integer

This method displays details on the analysis of the pullback line bundle of class l on a curve with components S.

### 3.4.6 AnalyzeBundleOnCurve (for IsList, IsList, IsInt)

▷ `AnalyzeBundleOnCurve(arg1, arg2, arg3)` (operation)

## 3.5 Analyse bundle on maximally degenerate curves

### 3.5.1 MaximallyDegenerateCurves (for IsList)

▷ `MaximallyDegenerateCurves(List, c)` (operation)

**Returns:** A list

This method identifies the maximal degenerations of a curve of class c in dP3.

### 3.5.2 EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves (for IsList, IsList)

▷ `EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves(Lists, c, l)` (operation)

**Returns:** A list

This method analysis the local and global sections of a pullback line bundle of class l on the maximally degenerate curves of class c.

### 3.5.3 EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves (for IsList, IsList, IsInt)

▷ EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves(*arg1*, *arg2*, *arg3*)  
(operation)

## 3.6 Examples

We can consider maximal degenerations of a given curve class and use these to estimate the number of global sections for a line bundle on this curve. This estimate is derived from counts of the local sections. Here is a simple example:

Example

```
gap> EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves(
> [ 3,-1,-1,-1 ], [ 1,-1,-3,-1 ] );;
```

For convenience, we allow the user to specify the level of detail from a verbose-integer as third argument. For example

Example

```
gap> EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves(
> [ 3,-1,-1,-1 ], [ 1,-1,-3,-1 ], 1 );;
```

```
Analyse bundle on 7 degenerate curves...
Estimated spectrum on 5 curves
Spectrum estimate: [ 2, 3 ]
```

The most details are provided for verbose level 2. Note that our counter assumes that neighbouring curve components do not support non-trivial sections simultaneously. This simplifies the estimate, but is a restrictive assumption at the same time. For example, in the following example, we cannot estimate a global section value at all from the maximal curve splits:

Example

```
gap> EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves(
> [ 4,-1,-2,-1 ], [ 3,-3,-1,-2 ], 1 );;
```

```
Analyse bundle on 10 degenerate curves...
Estimated spectrum on 0 curves
Spectrum estimate: [ ]
```

However, in other cases, we can estimate the number of global sections for all maximally degenerate curves:

Example

```
gap> EstimateGlobalSectionsOfBundleOnMaximallyDegenerateCurves(
> [ 5,-2,-2,-1 ], [ 2, -2, -4, -2 ] );
[ 0, 1, 2, 3, 4 ]
```

# Index

AnalyzeBundleOnCurve  
  for IsList, IsList, [9](#)  
  for IsList, IsList, IsInt, [9](#)

DegreesOfComponents  
  for IsList, IsToricVariety, [4](#)

DescendantLevel  
  for IsList, [4](#)

EstimateGlobalSections  
  for IsList, IsList, [9](#)

EstimateGlobalSectionsOfBundleOn-  
  MaximallyDegenerateCurves  
  for IsList, IsList, [9](#)  
  for IsList, IsList, IsInt, [10](#)

FindCounterBinary, [4](#)

FineApproximation  
  for IsList, IsList, [5](#)

FineApproximationWithSetups  
  for IsList, IsList, [5](#)

Genus  
  for IsList, [7](#)

IntersectionMatrix  
  for IsList, [8](#)

IntersectionNumber  
  for IsList, IsList, [7](#)

IntersectionsAmongCurveComponents  
  for IsList, [9](#)

IsE1Power  
  for IsList, [7](#)

IsE2Power  
  for IsList, [7](#)

IsE3Power  
  for IsList, [8](#)

IsE4Power  
  for IsList, [8](#)

IsE5Power  
  for IsList, [8](#)

IsE6Power  
  for IsList, [8](#)

IsIrreducible  
  for IsList, IsToricVariety, [4](#)

IsRigidPower  
  for IsList, [8](#)

IsSimpleSetup  
  for IsList, IsList, [9](#)

LineBundleDegree  
  for IsList, IsList, [7](#)

MaximallyDegenerateCurves  
  for IsList, [9](#)

RoughApproximation  
  for IsList, IsList, [5](#)

RoughApproximationWithSetups  
  for IsList, IsList, [5](#)

Sections  
  for IsInt, IsInt, [8](#)