Recent improvements to the numerical capabilities of ApCoCoA

Jan Limbeck

Chair of Symbolic Computation

Algebraic Oil Project Workshop Genova, 2009

Outline

Outline

Dealing With Complex Values
Application eigenvalues and vectors

- Used in one rational recovery algorithm
- Lapack returns the results of all computations in a non convenient way
- Behavior of all ApCoCoA functions like Num.EigenVectorsAndValues(...) has been changed to return both real and complex matrices

Dealing With Complex Values
Application eigenvalues and vectors

- Used in one rational recovery algorithm
- Lapack returns the results of all computations in a non convenient way
- Behavior of all ApCoCoA functions like Num.EigenVectorsAndValues(...) has been changed to return both real and complex matrices

Dealing With Complex Values
Application eigenvalues and vectors

- Used in one rational recovery algorithm
- Lapack returns the results of all computations in a non convenient way
- Behavior of all ApCoCoA functions like Num.EigenVectorsAndValues(...) has been changed to return both real and complex matrices

Outline

Port From CoCoAL To C++

Make all functions relevant to Risc available in the ApCoCoALib

- Calculation of the evaluation matrices
- Least squares fitting for the general problem $Ax \approx b$ using QR decomposition

Port From CoCoAL To C++

Make all functions relevant to Risc available in the ApCoCoALib

- Calculation of the evaluation matrices
- Least squares fitting for the general problem $Ax \approx b$ using QR decomposition

Least Squares Fitting

QR decomposition is a standard technique in numerical linear algebra

A compromise between speed and numerical stability

- Normal equations method is faster but numerically less stable
- truncated SVD method is slower but numerically "the best" solution

Let $M \in Mat_{m,n}(\mathbb{R})$, then we can decompose M in the following way

$$M = QR$$

with $Q \in Mat_m(\mathbb{R})$ an orthogonal matrix (i.e. $Q^T Q = I$) and $R \in Mat_{m,n}(\mathbb{R})$ an upper triangular matrix.

Least Squares Fitting

QR decomposition is a standard technique in numerical linear algebra

A compromise between speed and numerical stability

- Normal equations method is faster but numerically less stable
- truncated SVD method is slower but numerically "the best" solution

Let $M \in Mat_{m,n}(\mathbb{R})$, then we can decompose M in the following way

$$M = QR$$

with $Q \in Mat_m(\mathbb{R})$ an orthogonal matrix (i.e. $Q^T Q = I$) and $R \in Mat_{m,n}(\mathbb{R})$ an upper triangular matrix.

Least Squares Fitting

QR decomposition is a standard technique in numerical linear algebra

A compromise between speed and numerical stability

- Normal equations method is faster but numerically less stable
- truncated SVD method is slower but numerically "the best" solution

Let $M \in Mat_{m,n}(\mathbb{R})$, then we can decompose M in the following way

$$M = QR$$

with $Q \in Mat_m(\mathbb{R})$ an orthogonal matrix (i.e. $Q^T Q = I$) and $R \in Mat_{m,n}(\mathbb{R})$ an upper triangular matrix.

Least Squares Fitting

The idea

1. Calculate the QR decomposition of A using Lapack
2. Calculate R^{-1} for the non zero block of R
3. The least squares solution x is $R^{-1}Q^{Tr}b$

Outline

Improvement of the AVI polynomials

The AVI algorithm uses Stable Gauss to calculate the approximate border basis polynomials from the approximate kernel found by the SVD

Problem: The coefficients of the polynomials we obtain are not optimal with respect to the vanishing property. This is quite important in follow up computations like the Rational Recovery.

At the end of the algorithm correct the coefficients using least squares again.

The step is optional and can be activated from the CoCoA 4 GUI.

Improvement of the AVI polynomials

The AVI algorithm uses Stable Gauss to calculate the approximate border basis polynomials from the approximate kernel found by the SVD

Problem: The coefficients of the polynomials we obtain are not optimal with respect to the vanishing property. This is quite important in follow up computations like the Rational Recovery.

At the end of the algorithm correct the coefficients using least squares again.

The step is optional and can be activated from the CoCoA 4 GUI.

Improvement of the AVI polynomials

The AVI algorithm uses Stable Gauss to calculate the approximate border basis polynomials from the approximate kernel found by the SVD
Problem: The coefficients of the polynomials we obtain are not optimal with respect to the vanishing property. This is quite important in follow up computations like the Rational Recovery.
At the end of the algorithm correct the coefficients using least squares again.
The step is optional and can be activated from the CoCoA 4 GUI.

Implementation of the Sub Ideal BBM for Border Basis

An exact version of the Buchberger Möller algorithm for Border Basis has been implemented.

- basically the same general structure as for AVI
- switched from numerical computations with double data types to exact CoCoALib data types
- implementation of a naive algorithm to calculate the kernel of a matrix using the reduced row echelon form. There is still lots of room for future improvements.

Can be called with the same naming scheme from CoCoA4, only set epsilon to 0. Num.SubAVI(Points, 0, Ideal)

Implementation of the Sub Ideal BBM for Border Basis

An exact version of the Buchberger Möller algorithm for Border Basis has been implemented.

- basically the same general structure as for AVI
- switched from numerical computations with double data types to exact CoCoALib data types
- implementation of a naive algorithm to calculate the kernel of a matrix using the reduced row echelon form. There is still lots of room for future improvements.

Can be called with the same naming scheme from CoCoA4, only set epsilon to 0. Num.SubAVI(Points, 0, Ideal)

Implementation of the Sub Ideal BBM for Border Basis

An exact version of the Buchberger Möller algorithm for Border Basis has been implemented.

- basically the same general structure as for AVI
- switched from numerical computations with double data types to exact CoCoALib data types
- implementation of a naive algorithm to calculate the kernel of a matrix using the reduced row echelon form. There is still lots of room for future improvements.

Can be called with the same naming scheme from CoCoA4, only set epsilon to 0. Num.SubAVI(Points, 0, Ideal)

Implementation of the Sub Ideal BBM for Border Basis

An exact version of the Buchberger Möller algorithm for Border Basis has been implemented.

- basically the same general structure as for AVI
- switched from numerical computations with double data types to exact CoCoALib data types
- implementation of a naive algorithm to calculate the kernel of a matrix using the reduced row echelon form. There is still lots of room for future improvements.

Can be called with the same naming scheme from CoCoA4, only set epsilon to 0. Num.SubAVI(Points, 0, Ideal)

Implementation of the Sub Ideal AVI

Also for AVI a Sub Ideal Variant has been implemented.
Important addition as it allows to “feed in” some common knowledge about the problem into the algorithm.

Num.SubAVI(Points, 0.1, Ideal)

Implementation of the Sub Ideal AVI

Also for AVI a Sub Ideal Variant has been implemented.
Important addition as it allows to “feed in” some common knowledge about the problem into the algorithm.

Num.SubAVI(Points, 0.1, Ideal)

Implementation of the Sub Ideal AVI

Also for AVI a Sub Ideal Variant has been implemented.
Important addition as it allows to “feed in” some common knowledge about the problem into the algorithm.

Num.SubAVI(Points, 0.1, Ideal)

What lies ahead...

So AVI is finished?

There is still more to come...

In prepartion is a new variant of the AVI algorithm which does no longer need

- the correction of the coefficients of the polynomials to have a good fit
- clean up steps as they were introduced when moving from ABM to AVI
- a speed up which allows us the have a look at up to 50.000 points (not an issue at the moment but will become more important as we shift our focus towards exploration).

What lies ahead...

So AVI is finished?

There is still more to come...
In prepartion is a new variant of the AVI algorithm which does no longer need

- the correction of the coefficients of the polynomials to have a good fit
- clean up steps as they were introduced when moving from ABM to AVI
- a speed up which allows us the have a look at up to 50.000 points (not an issue at the moment but will become more important as we shift our focus towards exploration).

What lies ahead...

So AVI is finished?

There is still more to come...
In prepartion is a new variant of the AVI algorithm which does no longer need

- the correction of the coefficients of the polynomials to have a good fit
- clean up steps as they were introduced when moving from ABM to AVI
- a speed up which allows us the have a look at up to 50.000 points (not an issue at the moment but will become more important as we shift our focus towards exploration).

What lies ahead...

So AVI is finished?

There is still more to come...
In prepartion is a new variant of the AVI algorithm which does no longer need

- the correction of the coefficients of the polynomials to have a good fit
- clean up steps as they were introduced when moving from ABM to AVI
- a speed up which allows us the have a look at up to 50.000 points (not an issue at the moment but will become more important as we shift our focus towards exploration).

Outline

Algorithms For The Rational Recovery

Yesterday we heard about two quite different versions of the rational recovery algorithm:

- the eigenvalue method
- the "shape lemma" method

The first one has been implemented and will be released with the upcoming ApCoCoA 1.1 release.
A prototype of the second version has been implemented in the ApCoCoA library by Severin Neumann. Some issues are still remaining and need to be fixed for both versions.

Algorithms For The Rational Recovery

Yesterday we heard about two quite different versions of the rational recovery algorithm:

- the eigenvalue method
- the “shape lemma” method

The first one has been implemented and will be released with the upcoming ApCoCoA 1.1 release.
A prototype of the second version has been implemented in the ApCoCoA library by Severin Neumann. Some issues are still remaining and need to be fixed for both versions.

Algorithms For The Rational Recovery

Yesterday we heard about two quite different versions of the rational recovery algorithm:

- the eigenvalue method
- the “shape lemma” method

The first one has been implemented and will be released with the upcoming ApCoCoA 1.1 release.
A prototype of the second version has been implemented in the ApCoCoA library by Severin Neumann. Some issues are still remaining and need to be fixed for both versions.

Algorithms For The Rational Recovery
Usage

The eigenvalue method can be called via

Num.RatPoints(Polynomials, Order Ideal);

Experience shows that it works reasonably well in small and medium sized examples.

The best things come in threes

So another version of the rational recovery is in the works!

Idea: Use numerical optimization methods to iteratively improve the solution.

The algorithm we consider is the Fletcher-Reeves conjugate gradient method.

This is no exact method, that can find the global optimum, but a reasonable heuristic.

The best things come in threes

So another version of the rational recovery is in the works!
Idea: Use numerical optimization methods to iteratively improve the solution.
The algorithm we consider is the Fletcher-Reeves conjugate gradient method.
This is no exact method, that can find the global optimum, but a reasonable heuristic.

The best things come in threes

So another version of the rational recovery is in the works!
Idea: Use numerical optimization methods to iteratively improve the solution.
The algorithm we consider is the Fletcher-Reeves conjugate gradient method.
This is no exact method, that can find the global optimum, but a reasonable heuristic.

The best things come in threes

So another version of the rational recovery is in the works!
Idea: Use numerical optimization methods to iteratively improve the solution.
The algorithm we consider is the Fletcher-Reeves conjugate gradient method.
This is no exact method, that can find the global optimum, but a reasonable heuristic.

Some test results

First test results were very promising:

- Can process a full sized Shell example (about 2500 points) in 30 minutes. Still a lot room for improvements.
- Manged to improve an approximate border basis returned by AVI, which vanished approximately to 0.06 on the original set of points to 0.03 in a real world example!

Sorry for not being too specific today! More details next time.

Stay tuned!

Some test results

First test results were very promising:

- Can process a full sized Shell example (about 2500 points) in 30 minutes. Still a lot room for improvements.
- Manged to improve an approximate border basis returned by AVI, which vanished approximately to 0.06 on the original set of points to 0.03 in a real world example!

Sorry for not being too specific today! More details next time.

Stay tuned!

Some test results

First test results were very promising:

- Can process a full sized Shell example (about 2500 points) in 30 minutes. Still a lot room for improvements.
- Manged to improve an approximate border basis returned by AVI, which vanished approximately to 0.06 on the original set of points to 0.03 in a real world example!

Sorry for not being too specific today! More details next time.

Stay tuned!

Bug In Windows Port Of CoCoA 4

- CoCoA 4 produced always the same random numbers when using a finite field
- Only the windows version was affected
- Reason: Problem with the way threads are implemented in QT in windows....

Bug In Windows Port Of CoCoA 4

- CoCoA 4 produced always the same random numbers when using a finite field
- Only the windows version was affected
- Reason: Problem with the way threads are implemented in QT in windows....

Bug In Windows Port Of CoCoA 4

- CoCoA 4 produced always the same random numbers when using a finite field
- Only the windows version was affected
- Reason: Problem with the way threads are implemented in QT in windows....

Bug In Windows Port Of CoCoA 4

- Workaround: Start one thread in the beginning and feed in the data
- No need to invest more energy as the new Eclipse GUI reaches release status

Bug In Windows Port Of CoCoA 4

- Workaround: Start one thread in the beginning and feed in the data
- No need to invest more energy as the new Eclipse GUI reaches release status

Summary

- Better handling of complex values
- Promised functionality for Risc
- Improvements for AVI

- Outlook
  - A new variant of AVI
  - A new heuristic for a problem closely related to the rational recovery
  - Start with work on exploration

Summary

- Better handling of complex values
- Promised functionality for Risc
- Improvements for AVI

- Outlook
  - A new variant of AVI
  - A new heuristic for a problem closely related to the rational recovery
  - Start with work on exploration

Summary

- Better handling of complex values
- Promised functionality for Risc
- Improvements for AVI

- Outlook
  - A new variant of AVI
  - A new heuristic for a problem closely related to the rational recovery
  - Start with work on exploration

Summary

- Better handling of complex values
- Promised functionality for Risc
- Improvements for AVI

- Outlook
  - A new variant of AVI
  - A new heuristic for a problem closely related to the rational recovery
  - Start with work on exploration

Summary

- Better handling of complex values
- Promised functionality for Risc
- Improvements for AVI

- Outlook
  - A new variant of AVI
  - A new heuristic for a problem closely related to the rational recovery
  - Start with work on exploration

Summary

- Better handling of complex values
- Promised functionality for Risc
- Improvements for AVI

- Outlook
  - A new variant of AVI
  - A new heuristic for a problem closely related to the rational recovery
  - Start with work on exploration

Thanks for your patience

That's all folks!

Any questions?