The `broydensolve` package

Solve a system of equations with Broyden's good method

Matthias Floré

Version 1.0 (2025/07/20)

Abstract

This package implements Broyden's good method to solve a system of equations. It is also possible to use coordinates defined by Ti*k*Z as known and unknown variables.

Contents

1 Usage

The package `broydensolve` can be used by putting the following in the preamble.

```
\usepackage{broydensolve}
```

2 Motivation

To solve a system of (nonlinear) equations in a LaTeX document, for example `Python` with the package `pythontex` or `Sage` with the package `sagetex` can be used. This could require multiple compilations or the option `shell-escape`. The package `broydensolve` is implemented with LaTeX3, requires 1 compilation and does not require the option `shell-escape`. Moreover, it is possible to use coordinates defined by Ti*k*Z as known and unknown variables.

3 Commands

`\ang` ⋆ `\ang(`⟨*coordinate*⟩`)`
`\ang(`⟨*coordinate*$_1$⟩`,`⟨*coordinate*$_2$⟩`)`
`\ang(`⟨*coordinate*$_1$⟩`,`⟨*coordinate*$_2$⟩`,`⟨*coordinate*$_3$⟩`)`
`\ang(`⟨*coordinate*$_1$⟩`,`⟨*coordinate*$_2$⟩`,`⟨*coordinate*$_3$⟩`,`⟨*coordinate*$_4$⟩`)`

This command is defined while the command `\BroydenSolve` is executed. It can be used in the function(s) given to the key `func` and gives an angle in radians. For 1 coordinate, its expansion comes down to `atan(`$\langle y_1\rangle$`,`$\langle x_1\rangle$`)`, for 2 coordinates to `atan(`$\langle y_2\rangle$`-`$\langle y_1\rangle$`,`$\langle x_2\rangle$`-`$\langle x_1\rangle$`)`, for 3 coordinates to `atan(`$\langle y_3\rangle$`-`$\langle y_2\rangle$`,`$\langle x_3\rangle$`-`$\langle x_2\rangle$`)-atan(`$\langle y_1\rangle$`-`$\langle y_2\rangle$`,`$\langle x_1\rangle$`-`$\langle x_2\rangle$`)`, for 4 coordinates to `atan(`$\langle y_4\rangle$`-`$\langle y_3\rangle$`,`$\langle x_4\rangle$`-`$\langle x_3\rangle$`)-atan(`$\langle y_2\rangle$`-`$\langle y_1\rangle$`,`$\langle x_2\rangle$`-`$\langle x_1\rangle$`)` and if this is smaller than 0 then `2*pi` is added.

`\col` ⋆ `\col(`⟨*coordinate*$_1$⟩`,`⟨*coordinate*$_2$⟩`,`⟨*coordinate*$_3$⟩`)`

This command is defined while the command `\BroydenSolve` is executed. It can be used in the function(s) given to the key `func` and can be used to require that 3 points are collinear. Its expansion comes down to $\langle x_1\rangle(\langle y_2\rangle - \langle y_3\rangle) + \langle x_2\rangle(\langle y_3\rangle - \langle y_1\rangle) + \langle x_3\rangle(\langle y_1\rangle - \langle y_2\rangle)$.

`\dis` ⋆ `\dis(`⟨*coordinate*⟩`)`
`\dis(`⟨*coordinate*$_1$⟩`,`⟨*coordinate*$_2$⟩`)`

This command is defined while the command `\BroydenSolve` is executed. It can be used in the function(s) given to the key `func` and gives a distance. For 1 coordinate, its expansion comes down to $\sqrt{\langle x_1\rangle^2 + \langle y_1\rangle^2}$ and for 2 coordinates to $\sqrt{(\langle x_2\rangle - \langle x_1\rangle)^2 + (\langle y_2\rangle - \langle y_1\rangle)^2}$.

`\BroydenIterations` ⋆ `\BroydenIterations`

This command gives the total number of iterations after the command `\BroydenSolve` is executed. This command is expandable.

`\BroydenRoot` ⋆ `\BroydenRoot[`⟨*iteration*⟩`]{`⟨*variable*⟩`}`

This command gives the approximation for the root for ⟨*variable*⟩ after the command `\BroydenSolve` is executed. The default value for ⟨*iteration*⟩ is the total number of iterations. This command is expandable.

`\BroydenRoots` ⋆ `\BroydenRoots`

This command gives the approximation(s) for the root(s) for the variable(s) as a comma separated list after the command `\BroydenSolve` is executed. This command is expandable.

`\BroydenSetup` `\BroydenSetup{`⟨*keys*⟩`}`

This command sets the ⟨*keys*⟩ described in Section 4.

`\BroydenSolve` `\BroydenSolve{`⟨*keys*⟩`}`

This command sets the ⟨*keys*⟩ described in Section 4 inside a group and tries to approximate the root(s) for the function(s) defined by the key `func` with respect to the variable(s) defined by the key `var` with initial value(s) defined by the key `init` using Broyden's good method, see [1]. In general, this method does *not* always converge and does *not* give an exact result. Angles and trigonometric functions are typically best in radians for this purpose. The key `stop-crit` influences when the iteration stops.

4 Keys

`abs-approx-error` `abs-approx-error=`⟨*value*⟩

The ⟨*value*⟩ is used by the stopping criterion. Initially, it is `10^-3`.

`coordinates` `coordinates=`⟨*boolean*⟩

If true then the function variables are given by ⟨*variable*⟩`x` and ⟨*variable*⟩`y` for each ⟨*variable*⟩ in the ⟨*list*⟩ given to the key `var` until the number of variables is the same as the number of functions. The functions can contain names of coordinates defined by Ti*k*Z. For a coordinate ⟨*name*⟩, also ⟨*name*⟩`x` and ⟨*name*⟩`y` can be used in the functions to represent its x and y coordinate. This requires the package `tikz` and the Ti*k*Z library `calc`. After the iteration ends, Ti*k*Z coordinates are defined using the computed approximate solutions.

`func` `func=`⟨*list*⟩

The comma separated ⟨*list*⟩ defines the function(s). If a function contains a `,` then this function should be placed inside braces. Also a pair of braces can be required around the whole function(s).

`func-error` `func-error=`⟨*value*⟩

The ⟨*value*⟩ is used by the stopping criterion. Initially, it is `10^-3`.

init `init=`⟨*list*⟩

The ⟨*list*⟩ defines the initial value(s) for the iteration.

iterations `iterations=`⟨*number*⟩

The ⟨*number*⟩ is used by the stopping criterion. Initially, it is `5`.

rel-approx-error `rel-approx-error=`⟨*value*⟩

The ⟨*value*⟩ is used by the stopping criterion. Initially, it is `10^-3`.

stop-crit `stop-crit=abs-approx-error`
`stop-crit=func-error`
`stop-crit=iterations`
`stop-crit=rel-approx-error`

The stopping criterion influences when the iteration stops. If the function(s) evaluate(s) to 0 then the iteration stops immediately. Otherwise, the stopping behaviors are listed below.

`abs-approx-error` The iteration stops if the 1-norm of the difference between the current and the previous approximation is smaller than the value given to the key `abs-approx-error`.

`func-error` The iteration stops if the 1-norm of the function value(s) is smaller than the value given to the key `func-error`.

`iterations` The iteration stops after a number of iterations determined by the key `iterations`.

`rel-approx-error` The iteration stops if the 1-norm of the difference between the current and the previous approximation is smaller than the product of the value given to the key `rel-approx-error` and the 1-norm of the current approximation.

Initially, the stopping criterion is `rel-approx-error`.

var `var=`⟨*list*⟩

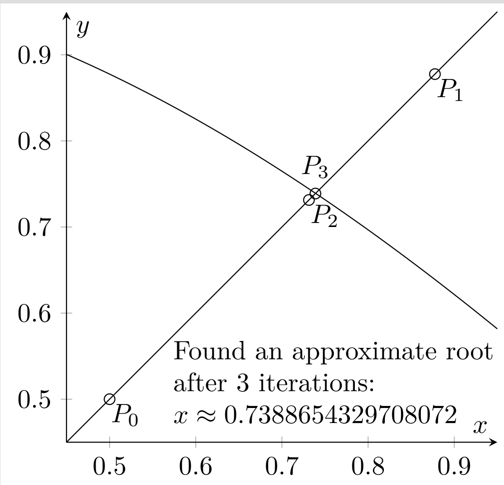
The ⟨*list*⟩ defines the function variable(s). Only the first N elements of the ⟨*list*⟩ are used where N is the number of functions. The variable(s) must consist entirely of Latin letters in the range `[a-zA-Z]` and cannot be an existing floating point identifier. Initially, it is `x,y,z`.

5 Examples

```
\BroydenSolve{
  func=x^5-x^2-1,
  init=1
}
Found an approximate root after \BroydenIterations{} iterations:\\
$x\approx\BroydenRoot{x}$
```

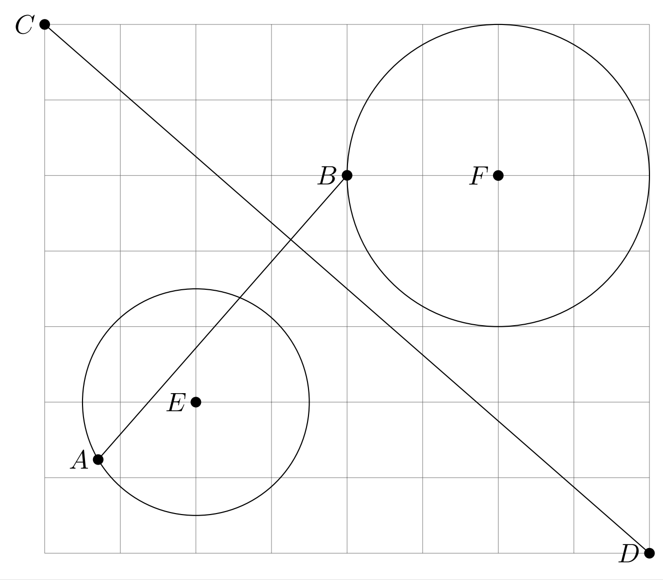
Found an approximate root after 8 iterations:
$x \approx 1.19385877093182$

```
%\usepackage{pgfplots}
%\pgfplotsset{compat=1.18}
\begin{tikzpicture}
\BroydenSolve{
  func=x-cos(x),
  init=0.5,
  rel-approx-error=10^-1
}
\begin{axis}[
  axis equal image,
  axis lines=middle,
  xmin=0.45,
  xmax=0.95,
  ymin=0.45,
  ymax=0.95,
  xlabel=$x$,
  ylabel=$y$
]
\addplot[smooth,domain=0.45:0.95,trig format plots=rad] {cos(x)};
\addplot[smooth] {x};
\pgfplotsinvokeforeach{0,...,\BroydenIterations}{
  \draw (\BroydenRoot[#1]{x},\BroydenRoot[#1]{x}) circle[radius=2pt]
    node[xshift=\fpeval{#1=3?0:6},yshift=\fpeval{#1=3?10:-6}] {$P_{#1}$};
}
\node[align=left] at (0.76,0.52) {Found an approximate root\\
  after \BroydenIterations{} iterations:\\$x\approx\BroydenRoot{x}$};
\end{axis}
\end{tikzpicture}
```

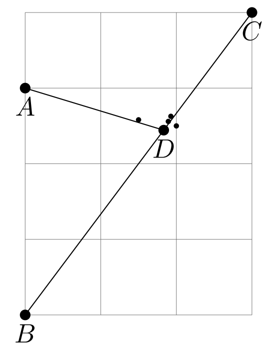


Below, we construct the points A and B so that A lies on the circle with center E, B lies on the circle with center F, $\|AB\| = 5$ and $AB \perp CD$.

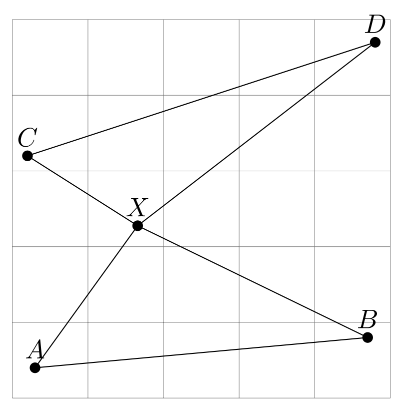
```
%\usepackage{tikz}
%\usetikzlibrary{calc}
\begin{tikzpicture}
\coordinate (C) at (-2,5);
\coordinate (D) at (6,-2);
\coordinate (E) at (0,0);
\coordinate (F) at (4,3);
\BroydenSolve{
  coordinates,
  func={
    \dis(A)-1.5,
    %{\dis(A,E)-1.5},
    %Ax^2+Ay^2-1.5^2,
    {\dis(B,F)-2},
    %(Bx-Fx)^2+(By-Fy)^2-2^2,
    {\dis(A,B)-5},
    {\ang(A,B,D,C)-pi/2}
    %{atan(Cy-Dy,Cx-Dx)-atan(By-Ay,Bx-Ax)-pi/2}
    %(Ax-Bx)*(Cx-Dx)+(Ay-By)*(Cy-Dy)
  },
  init={-1,-1,2,3},
  var={A,B}
}
\draw[help lines] (-2,-2) grid (6,5);
\draw (C)--(D) (E) circle[radius=1.5] (F) circle[radius=2] (A)--(B);
%(\BroydenRoot{Ax},\BroydenRoot{Ay})--(\BroydenRoot{Bx},\BroydenRoot{By})
\foreach\coord in {A,B,C,D,E,F}{
  \fill (\coord) circle[radius=2pt] node[left] {$\coord$};
}
\end{tikzpicture}
```



```
%\usepackage{tikz}
%\usetikzlibrary{calc}
\begin{tikzpicture}
\coordinate (A) at (0,3);
\coordinate (B) at (0,0);
\coordinate (C) at (3,4);
\BroydenSolve{
  coordinates,
  func={
    {\col(B,D,C)},
    {\ang(A,D,B)-70*deg}
  },
  init={2,2.5},
  var=D
}
\draw[help lines] (0,0) grid (3,4);
\draw (B)--(C) (A)--(D);
\foreach\n in {0,...,\inteval{\BroydenIterations-1}}{
  \fill (\BroydenRoot[\n]{Dx},\BroydenRoot[\n]{Dy}) circle[radius=1pt];
}
\foreach\coord in {A,B,C,D}{
  \fill (\coord) circle[radius=2pt] node[below] {$\coord$};
}
\end{tikzpicture}
```



```
%\usepackage{tikz}
%\usetikzlibrary{calc}
\begin{tikzpicture}
\coordinate (A) at (0.3,0.4);
\coordinate (B) at (4.7,0.8);
\coordinate (C) at (0.2,3.2);
\coordinate (D) at (4.8,4.7);
\BroydenSolve{
  coordinates,
  func={
    {\ang(A,X,B)-100*deg},
    {\ang(D,X,C)-110*deg}
  },
  init={2,2},
  var=X
}
\draw[help lines] (0,0) grid (5,5);
\draw (A)--(X)--(B)--cycle (C)--(X)--(D)--cycle;
\foreach\coord in {A,B,C,D,X}{
  \fill (\coord) circle[radius=2pt]
  node[above] {$\coord$};
}
\end{tikzpicture}
```



References

[1] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp. **19** (1965), 577-593.

Index

A The source code

```
%% broydensolve.sty
%% Copyright 2025 Matthias Floré
%
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3c
% of this license or (at your option) any later version.
% The latest version of this license is in
%   http://www.latex-project.org/lppl.txt
% and version 1.3c or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
%
% This work has the LPPL maintenance status `maintained'.
%
% The Current Maintainer of this work is Matthias Floré.
%
% This work consists of the files broydensolve-doc.pdf, broydensolve.sty,
% broydensolve-doc.tex and README.md.
\NeedsTeXFormat{LaTeX2e}
\ProvidesExplPackage{broydensolve}{2025/07/20}{1.0}{Solve a system of equations with Broyden's good method}
```

A.1 Variables

```
\bool_new:N \l__broydensolve_do_bool

\clist_new:N \g__broydensolve_roots_clist

\fp_new:N \l__broydensolve_aux_fp
\fp_new:N \g__broydensolve_coord_x_fp
\fp_new:N \g__broydensolve_coord_y_fp
\fp_new:N \l__broydensolve_det_fp
\fp_new:N \l__broydensolve_norm_h_fp
\fp_new:N \l__broydensolve_norm_x_fp

\int_new:N \g__broydensolve_iterations_int
\int_new:N \l__broydensolve_N_int
```

```
\seq_new:N \l__broydensolve_coordinates_seq
\seq_new:N \l__broydensolve_coords_seq
\seq_new:N \l__broydensolve_func_seq
\seq_new:N \l__broydensolve_init_seq
\seq_new:N \l__broydensolve_var_seq
\seq_new:N \l__broydensolve_var_N_seq

\tl_new:N \l__broydensolve_func_tl
\tl_new:N \l__broydensolve_name_tl
```

A.2 Keys

```
\keys_define:nn { broydensolve }
  {
    abs-approx-error .fp_set:N = \l__broydensolve_abs_approx_error_fp ,
    abs-approx-error .initial:n = 10^-3 ,
    coordinates .bool_set:N = \l__broydensolve_coordinates_bool ,
    func .tl_set:N = \l__broydensolve_funcs_tl ,
    func-error .fp_set:N = \l__broydensolve_func_error_fp ,
    func-error .initial:n = 10^-3 ,
    init .code:n = \exp_args:NNe \seq_set_from_clist:Nn \l__broydensolve_init_seq {#1} ,
    iterations .int_set:N = \l__broydensolve_iterations_int ,
    iterations .initial:n = 5 ,
    rel-approx-error .fp_set:N = \l__broydensolve_rel_approx_error_fp ,
    rel-approx-error .initial:n = 10^-3 ,
    stop-crit .str_set_e:N = \l__broydensolve_stop_crit_str ,
    stop-crit .initial:n = rel-approx-error ,
    var .code:n = \exp_args:NNe \seq_set_from_clist:Nn \l__broydensolve_var_seq {#1} ,
    var .initial:n = { x , y , z }
  }
```

A.3 Functions

```
\cs_new:Npn \__broydensolve_add:n #1
  { ( #1 < 0 ? ( #1 + 2 * pi ) : ( #1 ) ) }

\cs_new_protected:Npn \__broydensolve_add_name:N #1
```

```
{
  \tl_if_empty:nF {#1}
    {
      \cs_if_exist:cTF { __fp_parse_word_#1:N }
        { \tl_build_put_right:Nn \l__broydensolve_func_tl {#1} }
        {
          \bool_if:NTF \l__broydensolve_coordinates_bool
            {
              \tl_set:Ne \l__broydensolve_name_tl { \tl_range:nnn {#1} { 1 } { -2 } }
              \seq_if_in:NVTF \l__broydensolve_coordinates_seq \l__broydensolve_name_tl
                { \tl_build_put_right:Nn \l__broydensolve_func_tl { \cs:w l__broydensolve_var_#1_fp \cs_end: } }
                {
                  \seq_if_in:NVF \l__broydensolve_coords_seq \l__broydensolve_name_tl
                    {
                      \path let \p { l__broydensolve_coord } = ( \l__broydensolve_name_tl ) in
                        [
                          / utils / exec =
                            {
                              \fp_gset:Nn \g__broydensolve_coord_x_fp
                                {
                                  ( \pgf@yy * \x { l__broydensolve_coord } - \pgf@yx * \y { l__broydensolve_coord } )
                                  / \l__broydensolve_det_fp
                                }
                              \fp_gset:Nn \g__broydensolve_coord_y_fp
                                {
                                  ( \pgf@xx * \y { l__broydensolve_coord } - \pgf@xy * \x { l__broydensolve_coord } )
                                  / \l__broydensolve_det_fp
                                }
                            }
                        ] ;
                      \fp_zero_new:c { l__broydensolve_coord_\l__broydensolve_name_tl x_fp }
                      \fp_zero_new:c { l__broydensolve_coord_\l__broydensolve_name_tl y_fp }
                      \fp_set_eq:cN { l__broydensolve_coord_\l__broydensolve_name_tl x_fp } \g__broydensolve_coord_x_fp
                      \fp_set_eq:cN { l__broydensolve_coord_\l__broydensolve_name_tl y_fp } \g__broydensolve_coord_y_fp
                      \seq_put_right:NV \l__broydensolve_coords_seq \l__broydensolve_name_tl
                    }
                  \tl_build_put_right:Ne \l__broydensolve_func_tl { \fp_use:c { l__broydensolve_coord_#1_fp } }
                }
            }
```

```
          { \tl_build_put_right:Nn \l__broydensolve_func_tl { \cs:w l__broydensolve_var_#1_fp \cs_end: } }
        }
    }
  }

\cs_new:Npn \__broydensolve_atan:nnn #1#2#3
  { atan ( \clist_item:nn {#1} {#2} y - \clist_item:nn {#1} {#3} y , \clist_item:nn {#1} {#2} x - \clist_item:nn {#1} {#3} x ) }

\cs_new_protected:Npn \__broydensolve_init:nn #1#2
  {
    \fp_gzero_new:c { g__broydensolve_x_0_#2_fp }
    \fp_gset:cn { g__broydensolve_x_0_#2_fp } {#1}
    \fp_zero_new:c { l__broydensolve_var_#2_fp }
    \fp_set:cn { l__broydensolve_var_#2_fp } {#1}
  }

\cs_new_protected:Npn \__broydensolve_stop_crit:
  {
    \fp_set:Nn \l__broydensolve_aux_fp { abs ( \seq_use:Nn \l__broydensolve_func_seq { ) + abs ( } ) }
    \fp_compare:nNnTF { \l__broydensolve_aux_fp } > { 0 }
      {
        \str_case:VnF \l__broydensolve_stop_crit_str
          {
            { abs-approx-error }
              {
                \int_if_zero:nF { \g__broydensolve_iterations_int }
                  {
                    \fp_zero:N \l__broydensolve_norm_h_fp
                    \int_step_inline:nn { \l__broydensolve_N_int }
                      { \fp_add:Nn \l__broydensolve_norm_h_fp { abs ( \cs:w l__broydensolve_h_##1_fp \cs_end: ) } }
                    \bool_set:Nn \l__broydensolve_do_bool
                      { \fp_compare_p:nNn { \l__broydensolve_norm_h_fp } < { \l__broydensolve_abs_approx_error_fp } }
                  }
              }
            { func-error }
              {
                \bool_set:Nn \l__broydensolve_do_bool
                  { \fp_compare_p:nNn { \l__broydensolve_aux_fp } < { \l__broydensolve_func_error_fp } }
```

```
       }
      { iterations }
       {
        \bool_set:Nn \l__broydensolve_do_bool
         { \int_compare_p:nNn { \g__broydensolve_iterations_int } = { \l__broydensolve_iterations_int } }
       }
      { rel-approx-error }
       {
        \int_if_zero:nF { \g__broydensolve_iterations_int }
         {
          \fp_zero:N \l__broydensolve_norm_h_fp
          \fp_zero:N \l__broydensolve_norm_x_fp
          \seq_map_indexed_inline:Nn \l__broydensolve_var_N_seq
           {
            \fp_add:Nn \l__broydensolve_norm_h_fp { abs ( \cs:w l__broydensolve_h_##1_fp \cs_end: ) }
            \fp_add:Nn \l__broydensolve_norm_x_fp { abs ( \cs:w l__broydensolve_var_##2_fp \cs_end: ) }
           }
          \bool_set:Nn \l__broydensolve_do_bool
           {
            \fp_compare_p:nNn
             { \l__broydensolve_norm_h_fp } < { \l__broydensolve_rel_approx_error_fp * \l__broydensolve_norm_x_fp }
           }
         }
       }
     }
     { \PackageError { broydensolve } { Wrong~value~for~key~stop-crit } {} }
   }
   { \bool_set_true:N \l__broydensolve_do_bool }
 }
```

A.4 Document commands

```
\NewExpandableDocumentCommand \BroydenIterations {}
 { \int_use:N \g__broydensolve_iterations_int }

\NewExpandableDocumentCommand \BroydenRoot { O { \g__broydensolve_iterations_int } m }
 { \fp_use:c { g__broydensolve_x_\int_eval:n {#1}_#2_fp } }
```

```
\NewExpandableDocumentCommand \BroydenRoots {}
  { \g__broydensolve_roots_clist }

\NewDocumentCommand \BroydenSetup { m }
  { \keys_set:nn { broydensolve } {#1} }

\NewDocumentCommand \BroydenSolve { m }
  {
    \group_begin:
      \keys_set:nn { broydensolve } {#1}
      \DeclareExpandableDocumentCommand \ang { r () }
        {
          \__broydensolve_add:n
            {
              \int_case:nnF { \clist_count:n {##1} }
                {
                  { 1 } { atan ( \clist_item:nn {##1} { 1 } y , \clist_item:nn {##1} { 1 } x ) }
                  { 2 } { \__broydensolve_atan:nnn {##1} { 2 } { 1 } }
                  { 3 } { \__broydensolve_atan:nnn {##1} { 3 } { 2 } - \__broydensolve_atan:nnn {##1} { 1 } { 2 } }
                  { 4 } { \__broydensolve_atan:nnn {##1} { 4 } { 3 } - \__broydensolve_atan:nnn {##1} { 2 } { 1 } }
                }
                { \ang {} }
            }
        }
      \DeclareExpandableDocumentCommand \col { r () }
        {
          \int_compare:nNnTF { \clist_count:n {##1} } = { 3 }
            {
              \clist_item:nn {##1} { 1 } x * ( \clist_item:nn {##1} { 2 } y - \clist_item:nn {##1} { 3 } y )
              + \clist_item:nn {##1} { 2 } x * ( \clist_item:nn {##1} { 3 } y - \clist_item:nn {##1} { 1 } y )
              + \clist_item:nn {##1} { 3 } x * ( \clist_item:nn {##1} { 1 } y - \clist_item:nn {##1} { 2 } y )
            }
            { \col {} }
        }
      \DeclareExpandableDocumentCommand \dis { r () }
        {
          \int_case:nnF { \clist_count:n {##1} }
            {
```

```
        { 1 } { sqrt ( \clist_item:nn {##1} { 1 } x ^ 2 + \clist_item:nn {##1} { 1 } y ^ 2 ) }
        { 2 }
          {
            sqrt
              (
                ( \clist_item:nn {##1} { 2 } x - \clist_item:nn {##1} { 1 } x ) ^ 2
                + ( \clist_item:nn {##1} { 2 } y - \clist_item:nn {##1} { 1 } y ) ^ 2
              )
          }
      }
      { \dis {} }
  }
\int_set:Nn \l__broydensolve_N_int { \clist_count:e { \l__broydensolve_funcs_tl } }
\bool_if:NTF \l__broydensolve_coordinates_bool
  {
    \seq_map_indexed_inline:Nn \l__broydensolve_var_seq
      {
        \seq_put_right:Nn \l__broydensolve_var_N_seq { ##2 x }
        \seq_put_right:Nn \l__broydensolve_var_N_seq { ##2 y }
        \seq_put_right:Nn \l__broydensolve_coordinates_seq {##2}
        \int_compare:nNnT {##1} = { \l__broydensolve_N_int / 2 }
          { \seq_map_break: }
      }
    \fp_set:Nn \l__broydensolve_det_fp { \pgf@yy * \pgf@xx - \pgf@yx * \pgf@xy }
  }
  {
    \seq_map_indexed_inline:Nn \l__broydensolve_var_seq
      {
        \seq_put_right:Nn \l__broydensolve_var_N_seq {##2}
        \int_compare:nNnT {##1} = { \l__broydensolve_N_int }
          { \seq_map_break: }
      }
  }
\exp_args:Ne \clist_map_inline:nn { \l__broydensolve_funcs_tl }
  {
    \tl_build_begin:N \l__broydensolve_func_tl
    \tl_build_begin:N \l__broydensolve_name_tl
    \tl_map_inline:nn {##1}
      {
```

```
        \bool_lazy_and:nnTF
          { \bool_lazy_or_p:nn { \int_compare_p:nNn { `####1 } < { `a } } { \int_compare_p:nNn { `z } < { `####1 } } }
          { \bool_lazy_or_p:nn { \int_compare_p:nNn { `####1 } < { `A } } { \int_compare_p:nNn { `Z } < { `####1 } } }
          {
            \tl_build_end:N \l__broydensolve_name_tl
            \exp_args:NV \__broydensolve_add_name:N \l__broydensolve_name_tl
            \tl_build_begin:N \l__broydensolve_name_tl
            \tl_build_put_right:Nn \l__broydensolve_func_tl {####1}
          }
          { \tl_build_put_right:Nn \l__broydensolve_name_tl {####1} }
      }
    \tl_build_end:N \l__broydensolve_name_tl
    \exp_args:NV \__broydensolve_add_name:N \l__broydensolve_name_tl
    \tl_build_end:N \l__broydensolve_func_tl
    \seq_put_right:NV \l__broydensolve_func_seq \l__broydensolve_func_tl
  }
\seq_map_pairwise_function:NNN \l__broydensolve_init_seq \l__broydensolve_var_N_seq \__broydensolve_init:nn
\int_step_inline:nn { \l__broydensolve_N_int }
  {
    \int_step_inline:nn { \l__broydensolve_N_int }
      { \fp_zero_new:c { l__broydensolve_B_##1_####1_fp } }
    %set B to the identity matrix
    \fp_set:cn { l__broydensolve_B_##1_##1_fp } { 1 }
    \fp_zero_new:c { l__broydensolve_w_##1_fp }
    \fp_zero_new:c { l__broydensolve_h_##1_fp }
    \fp_zero_new:c { l__broydensolve_aux_##1_fp }
    \fp_zero_new:c { l__broydensolve_auxi_##1_fp }
  }
\int_gzero:N \g__broydensolve_iterations_int
\__broydensolve_stop_crit:
\bool_until_do:Nn \l__broydensolve_do_bool
  {
    %set w=-f(x)
    \seq_map_indexed_inline:Nn \l__broydensolve_func_seq
      { \fp_set:cn { l__broydensolve_w_##1_fp } { - (##2) } }
    %set h=-B*f(x)=B*w
    \int_step_inline:nn { \l__broydensolve_N_int }
      {
        \fp_zero:c { l__broydensolve_h_##1_fp }
```

```
    \int_step_inline:nn { \l__broydensolve_N_int }
      {
        \fp_add:cn { l__broydensolve_h_##1_fp }
          { \cs:w l__broydensolve_B_##1_####1_fp \cs_end: * \cs:w l__broydensolve_w_####1_fp \cs_end: }
      }
  }
%set the variable(s) to x+h
\seq_map_indexed_inline:Nn \l__broydensolve_var_N_seq
  { \fp_add:cn { l__broydensolve_var_##2_fp } { \cs:w l__broydensolve_h_##1_fp \cs_end: } }
%add f(x+h) to w so that w=f(x+h)-f(x)
\seq_map_indexed_inline:Nn \l__broydensolve_func_seq
  { \fp_add:cn { l__broydensolve_w_##1_fp } {##2} }
%update B
\int_step_inline:nn { \l__broydensolve_N_int }
  {
    \fp_zero:c { l__broydensolve_aux_##1_fp }
    \int_step_inline:nn { \l__broydensolve_N_int }
      {
        \fp_add:cn { l__broydensolve_aux_##1_fp }
          { \cs:w l__broydensolve_B_##1_####1_fp \cs_end: * \cs:w l__broydensolve_w_####1_fp \cs_end: }
      }
  }
\fp_zero:N \l__broydensolve_aux_fp
\int_step_inline:nn { \l__broydensolve_N_int }
  {
    \fp_add:Nn \l__broydensolve_aux_fp { \cs:w l__broydensolve_h_##1_fp \cs_end: * \cs:w l__broydensolve_aux_##1_fp \cs_end: }
    \fp_sub:cn { l__broydensolve_aux_##1_fp } { \cs:w l__broydensolve_h_##1_fp \cs_end: }
  }
\int_step_inline:nn { \l__broydensolve_N_int }
  {
    \fp_zero:c { l__broydensolve_auxi_##1_fp }
    \int_step_inline:nn { \l__broydensolve_N_int }
      {
        \fp_add:cn { l__broydensolve_auxi_##1_fp }
          { \cs:w l__broydensolve_h_####1_fp \cs_end: * \cs:w l__broydensolve_B_####1_##1_fp \cs_end: }
      }
  }
\int_step_inline:nn { \l__broydensolve_N_int }
  {
```

```
            \int_step_inline:nn { \l__broydensolve_N_int }
              {
                \fp_sub:cn { l__broydensolve_B_##1_####1_fp }
                  {
                    \cs:w l__broydensolve_aux_##1_fp \cs_end: * \cs:w l__broydensolve_auxi_####1_fp \cs_end:
                    / \l__broydensolve_aux_fp
                  }
              }
          }
        \int_gincr:N \g__broydensolve_iterations_int
        \seq_map_inline:Nn \l__broydensolve_var_N_seq
          {
            \fp_gzero_new:c { g__broydensolve_x_\int_use:N \g__broydensolve_iterations_int _##1_fp }
            \fp_gset_eq:cc { g__broydensolve_x_\int_use:N \g__broydensolve_iterations_int _##1_fp } { l__broydensolve_var_##1_fp }
          }
        \__broydensolve_stop_crit:
      }
    \clist_gclear:N \g__broydensolve_roots_clist
    \seq_map_inline:Nn \l__broydensolve_var_N_seq
      { \clist_gput_right:Ne \g__broydensolve_roots_clist { \BroydenRoot {##1} } }
    \bool_if:NT \l__broydensolve_coordinates_bool
      {
        \seq_map_inline:Nn \l__broydensolve_coordinates_seq
          { \coordinate (##1) at ( \BroydenRoot { ##1 x } , \BroydenRoot { ##1 y } ) ; }
      }
  \group_end:
}

\endinput
```