# Atlas Algorithms

Harry Dankowicz
Department of Mechanical Science and Engineering
University of Illinois at Urbana-Champaign

March 22, 2020

## Contents

# 1 Introduction

A COCO-compatible paradigm for "parameter continuation" is described in *Recipes for Continuation*[1], specifically the parts on "Atlas Algorithms" and "Event Handling".

Additionally, the March 2020 release includes the production-ready `'atlas_kd'` atlas algorithm for multidimensional solution manifolds for adaptive continuation problems (with varying embedding dimension and variable interpretation)[2]. This is described in detail in this tutorial and exemplified using demos in the `coco/covering/examples` folder in the COCO release.

# 2 Principles of continuation

The COCO platform supports the construction of continuous families of approximate solutions to operator equations in inner-product spaces. Such approximate solutions lie in finite-dimensional restrictions of the operator domain and satisfy constraints that ensure consistency and convergence toward some exact solution in a suitable limit. No limit process is necessary in *algebraic* problems, defined on finite-dimensional vector spaces.

The successive growth of the family of solutions is known as *continuation*. Continuation proceeds in stages of *expansion* and *consolidation*. During expansion, new solutions are constructed from, and in the vicinity of, existing solutions. During consolidation, new solutions are integrated with existing solutions, thereby establishing opportunities for further expansion that avoid treading over familiar ground.

In infinite-dimensional problems, consecutive stages of expansion may be formulated on restrictions of different finite dimension, as well as with different interpretations of the problem variables and different numbers of problem constraints. Such variations may be necessary in order to ensure a desired accuracy of approximation in all parts of the operator domain. In this case, we say that the problem construction is *adaptive*. A typical example is the use of a varying mesh for the discretization of a trajectory continuation problem. For adaptive problems, COCO requires that the difference between the number of problem variables and number of problem constraints remains constant throughout continuation.

The process of continuation is seeded with an *initial solution guess* and a single auxiliary *projection condition* that, nominally, identifies a locally unique solution. The latter is found through an iterative search using a *corrector*, e.g., Newton's method. In the simplest case, the projection condition is linear in the deviation of the unknowns from the initial solution guess.

In COCO, each stage of expansion is similarly seeded with a *predictor* and a one-parameter family of projection conditions that identify, pointwise one-to-one, a locally unique and con-

[1] Dankowicz, H. & Schilder, F., *Recipes for Continuation*, Society for Industrial and Applied Mathematics, 2013.

[2] Dankowicz, H., Wang, Y., Schilder, F. & Henderson, M.E., "Multidimensional Manifold Continuation for Adaptive Boundary-Value Problems," *J. Computational and Nonlinear Dynamics* **15(5)**, art. no. 051002, 2020.

tinuous curve of solutions that includes the existing solution used to construct the predictor. A single stage of expansion may produce multiple new solutions along such a curve. For example, solutions may be located that correspond to particular values of monitor functions that vary continuously along the curve.

A stage of expansion is *properly resolved* if the new solutions located along the corresponding curve are representative of all solutions on the curve segment between the existing solution and the furthermost new solution in either direction. In this case, consolidation should ensure that no further attempts are made to locate new solutions on this curve segment.

Each run of the `coco` entry-point function is associated with an *atlas algorithm* that implements specific solutions to the tasks of expansion and consolidation. Atlas algorithms may differ in the formulation of projection conditions, in the construction of a predictor, in the deployment of a corrector, in the definition of what it means for a stage of expansion to be properly resolved, and in the actions taken when this is not the case. Some atlas algorithms support adaptive problem constructions, others don't. Some atlas algorithms can handle multi-dimensional manifolds of solutions, others are restricted to the one-dimensional case. Some may be able to detect special points associated with local properties of the family of solutions like fold points and branch points, others may not. Some atlas algorithms are designed for particular classes of problems, others are general-purpose and therefore inevitably less efficient for some particular problems.

The naming of atlas algorithms follows the convention that an identifying string is appended to the string `'atlas_'`. To use an atlas algorithm named `'atlas_alg'` in a run of the `coco` entry-point function applied to the continuation problem structure `prob`, use the syntax

```
>> prob = coco_set(prob, 'cont', 'atlas', 'alg');
```

before a subsequent call to `coco`. In the absence of such an assignment, `coco` uses the default atlas algorithm `'atlas_1d'` for one-dimensional solution manifolds and `'atlas_kd'` for higher-dimensional solution manifolds. To set options for a selected atlas algorithm, use the syntax

```
>> prob = coco_set(prob, 'cont', ...
```

where the ellipses denote pairs of settings and values specific to the algorithm. All atlas algorithms support options defining the frequency of screen outputs (`'NPR'` with default value 10) and the frequency of storing solutions to disk (`'NSV'` with default value equal to the value of `'NPR'`). Additional options (`'corrector'` and `'linsolve'` with default values `'nwtn'` and `'splu'`) allow for identifying the nonlinear and linear solvers used by the corrector. For a list of supported settings and their default values for an atlas algorithm named `'atlas_alg'`, see the output of the `atlas_alg_settings` utility, when available.

# 3 Default atlas algorithms

The COCO default atlas algorithms are designed to support general problem constructions, including composite continuation problems with adaptive problem structure. In the absence of any assumed problem form or meaning of the problem variables, these default atlas algorithms are ignorant of the properties of the operator domain and the associated inner product and (induced) geometry. Nevertheless, as explained in the case of `'atlas_kd'` below, nontrivial geometric analysis is possible by clever use of monitor functions during problem construction.

## 3.1 One-dimensional manifolds: `'atlas_1d'`

The `'atlas_1d'` atlas algorithm encodes an *expanding boundary algorithm* for solution manifolds of dimension one. The algorithm is able to continue in both directions along the solution manifold from an initial point, and can also start on the boundary of a computational domain as long as the solution manifold intersects the boundary transversally at the initial point. The algorithm is able to detect fold points in a selected active continuation parameter, as well as branch points associated with a singular problem Jacobian. Except for special points, the algorithm constructs only one new solution in each stage of expansion.

In the `'atlas_1d'` atlas algorithm, the scalar-valued projection condition that accompanies an initial solution guess is linear and homogeneous in the deviation from the initial solution guess. Provided that the initial solution guess is close to a regular solution (where the problem Jacobian is nonsingular), the coefficient vector equals the unit vector along the first active continuation parameter axis in the direction of increasing values of the continuation parameter, unless this vector is close to perpendicular to the solution manifold at the nearby solution. In the latter case, the coefficient vector approximates a unit length tangent vector to the solution manifold at the nearby solution. Continuation proceeds first in the direction opposite to the coefficient vector. If the initial solution guess is close to a branch point, the `'t0'` flag may be used in a call to `coco_add_func` in order to force construction of an approximate tangent vector along one of the intersecting branches. In this case, no corrector is applied to the initial solution guess.

In each stage of expansion, the `'atlas_1d'` algorithm relies on i) a scalar-valued pseudo-arclength projection condition defined in terms of a tangent vector to the solution manifold at an existing solution and ii) a step size parameter. The algorithm supports adaptive step size adjustments along the solution manifold in order to achieve properly resolved stages of expansion with a reasonably small set of individual solutions. Here, the step size associated with a particular stage of expansion is described locally in terms of the Euclidean distance between an existing solution and the predictor constructed along a tangent line to the solution manifold. The algorithm does not support global measurements of solution similarity and, consequently, is unable to avoid redundant coverage of closed solution manifolds.

The `'atlas_1d'` atlas algorithm supports adaptive problem constructions, including changes to the numbers and interpretations of problem variables and problem constraints. Such adaptive updates are applied before the construction of a predictor in a stage of expan-

sion. Indeed, after each such adaptive update, the corrector is first applied with zero step size in order to construct a corrected solution on the solution manifold and a local tangent vector in the new problem variable domain. The predictor and projection condition are constructed from this corrected solution and tangent vector.

Since a nonzero step size is associated with a Euclidean distance in the problem variable domain, it lacks invariant meaning during analysis of a continuation problem with adaptive problem construction. For a given step size, a significant increase in the number of problem variables may result in significantly smaller variations in any continuation parameters. In a worst-case scenario, such variations might fall below the tolerance triggering event detection.

The `'atlas_1d'` atlas algorithm supports options triggering the detection of fold points (`'FP'` which is true by default) and branch points (`'BP'` which is true by default), as well as an optional string label of an active continuation parameter (`'fpar'`) associated with fold point detection (which, otherwise, defaults to the first active continuation parameter). Optional settings governing adaptive problem construction include the frequency of adaptive updates (`'NAdapt'` with default value of 0, i.e., no adaptive updates) and the maximum number of supported sweeps through toolbox remeshing routines during each update (`'RMMX'` with default value of 10).

Adaptive step-size management in `'atlas_1d'` is governed by optional settings defining the initial step size (`'h0'` with default value 0.1), the maximum step size (`'h_max'` with default value 0.5), the minimum step size (`'h_min'` with default value 0.01), the maximum step size adaptation factor (`'h_fac_max'` with default value 2.0), and the minimum step size adaptation factor (`'h_fac_min'` with default value 0.5). Additional relevant optional settings include the maximum Euclidean norm of the problem residual evaluated at a predictor (`'MaxRes'` with default value 0.1) and the maximum angle between two consecutive tangent vectors on a curve segment (`'al_max'` with default value 7 degrees). As suggested above, the interpretation of these settings lacks invariance in continuation problems with adaptive problem construction. Some experimentation with the settings may be required in order to generate satisfactory output in such problems.

Continuation along a one-dimensional solution manifold with `'atlas_1d'` terminates once the computational boundary is reached from an interior initial point, or once the maximum number of steps along each direction is reached. The latter is determined by the optional setting `'PtMX'` which defaults to 100, corresponding to 100 steps in each direction. An assignment

```
>> prob = coco_set(prob, 'cont', 'PtMX', [ neg, pos ]);
```

for some nonnegative integers `neg` and `pos` results in `neg` steps in the initial direction of continuation and `pos` steps in the opposite direction.

## 3.2 Multi-dimensional manifolds: `'atlas_kd'`

The `'atlas_kd'` atlas algorithm encodes an expanding boundary algorithm for solution manifolds of dimension $d$ greater than or equal to one provided that the number of active continuation parameters is greater than or equal to $d$. The algorithm generates an approx-

imate polyhedral tessellation of the solution manifold that grows from an initial point, and can start on the boundary of a computational domain as long as the solution manifold intersects the boundary transversally at the initial point. Special points may be detected only along individual curve segments. Except for special points, the algorithm constructs only one new solution in each stage of expansion. Each stage of consolidation extends the polyhedral tessellation and assigns individual polyhedra to distinct collections of boundary charts, interior charts, and charts on the boundary of the computational domain.

In the `'atlas_kd'` atlas algorithm, the projection condition that accompanies an initial solution guess is linear and homogeneous in the deviation from the initial solution guess. Provided that the initial solution guess is close to a regular solution, the $d$ rows of the coefficient matrix constitute an orthonormal basis for a space that includes unit vectors along the first ($\leq d$) active continuation parameter axes, except for those that are close to perpendicular to the solution manifold at the nearby solution. If the initial solution guess is close to a branch point, the `'t0'` flag may be used in a call to `coco_add_func` in order to force construction of an orthonormal basis of approximate tangent vectors to one of the intersecting hypersurfaces. In this case, no corrector is applied to the initial solution guess.

In each stage of expansion, the `'atlas_kd'` algorithm relies on i) a pseudo-arclength projection condition defined in terms of a tangent matrix whose $d$ rows constitute an orthonormal basis of the tangent space to the solution manifold associated with a boundary chart, ii) a unit direction vector, and iii) a step size parameter. The algorithm supports adaptive step size adjustments along the solution manifold in order to achieve properly resolved stages of expansion with a reasonably small set of individual solutions. Here, the step size associated with a particular stage of expansion is described locally in terms of the Euclidean distance in the active continuation parameter domain between an existing solution and the predictor constructed along a tangent line to the solution manifold. The algorithm supports global measurements of solution similarity in this domain and, consequently, is able to avoid redundant coverage of closed solution manifolds.

The `'atlas_kd'` atlas algorithm supports adaptive problem constructions, including changes to the numbers and interpretations of problem variables and problem constraints. Such adaptive updates are applied before the construction of a predictor in a stage of expansion. Indeed, after each such adaptive update, the corrector is first applied with zero step size in order to construct a corrected solution on the solution manifold and a local tangent vector in the new problem variable domain. The predictor and projection condition are constructed from this corrected solution and tangent vector.

Since a nonzero step size is associated with a Euclidean distance in the active continuation parameter domain, it has invariant meaning during analysis of a continuation problem with adaptive problem construction provided that the corresponding monitor functions are invariantly defined.

Optional settings of the `'atlas_kd'` atlas algorithm governing adaptive problem construction include the frequency of adaptive updates (`'NAdapt'` with default value of 0) associated with any chain of successively constructed solutions and the maximum number of supported sweeps through toolbox remeshing routines during each update (`'RMMX'` with default value

of 10).

Adaptive step-size management in `'atlas_kd'` is governed by optional settings defining the initial step size (`'R'` with default value 0.1), the maximum step size (`'R_max'` with default value 1), the minimum step size (`'R_min'` with default value 0.00001), the maximum step size adaptation factor (`'R_fac_max'` with default value 2.0), and the minimum step size adaptation factor (`'R_fac_min'` with default value 0.5). Additional relevant optional settings include the maximum Euclidean norm of the problem residual evaluated at a predictor (`'MaxRes'` with default value 1) and the maximum angle between two consecutive tangent vectors on a curve segment (`'al_max'` with default value 10 degrees).

Continuation along a one-dimensional solution manifold with `'atlas_kd'` terminates once there are no more boundary charts, or once the maximum number of steps is reached. The latter is determined by the optional setting `'PtMX'` which defaults to 100.

The `'atlas_kd'` atlas algorithm stores a cell array consisting of all computed charts and a list of boundary chart integer identifiers in the `atlas` field of the bifurcation data cell array that is stored to disk after each successful continuation step. This information may be used to visualize the polyhedral tessellation using the `plot_atlas_kd` utility.

# 4 Application – `linode`

Consider the linear oscillator with harmonic excitation governed by the explicitly time-dependent dynamical system

$$\dot{x}_1 = x_2,\ \dot{x}_2 = -p_1 x_1 - x_2 + p_2 \cos t \tag{1}$$

in terms of the vector of state variables $x = (x_1, x_2) \in \mathbb{R}^2$ and the vector of problem parameters $p = (p_1, p_2) \in \mathbb{R}^2$. For arbitrary initial conditions, the steady-state behavior is then given by the $2\pi$-periodic orbit

$$x_1(t) = x_1^*(t) := p_2 \frac{\sin t + (p_1 - 1)\cos t}{p_1^2 - 2p_1 + 2},\ x_2(t) = x_2^*(t) := p_2 \frac{\cos t - (p_1 - 1)\sin t}{p_1^2 - 2p_1 + 2}. \tag{2}$$

We consider below continuation along this family of periodic orbits under variations only in $p_1$ or simultaneously in $p_1$ and $p_2$.

We first invoke the `ode_isol2coll` constructor and append periodic boundary conditions with $T = 2\pi$ and $T_0 = 0$, as shown in the commands below.

```
>> prob = coco_prob;
>> prob = coco_set(prob, 'ode', 'autonomous', false);
>> prob = coco_set(prob, 'coll', 'NTST', 10, 'MXCL', false);
>> p0   = [1; 1];
>> t0   = (0:2*pi/99:2*pi)';
>> x0   = [p0(2)*(sin(t0)+(p0(1)-1)*cos(t0))/(p0(1)^2-2*p0(1)+2) ...
     p0(2)*(cos(t0)-(p0(1)-1)*sin(t0))/(p0(1)^2-2*p0(1)+2) ];
>> prob = ode_isol2coll(prob, '', @linode, t0, x0, {'p1' 'p2'}, p0);
>> [data, uidx] = coco_get_func_data(prob, 'coll', 'data', 'uidx');
>> maps = data.coll_seg.maps;
>> prob = coco_add_func(prob, 'po', @linode_bc, [], 'zero', ...
     'uidx', uidx([maps.x0_idx; maps.x1_idx; maps.T0_idx; maps.T_idx]));
```

where

```
function y = linode(t, x, p)

x1 = x(1,:);
x2 = x(2,:);
p1 = p(1,:);
p2 = p(2,:);

y(1,:) = x2;
y(2,:) = -x2-p1.*x1+p2.*cos(t);

end

function [data, y] = linode_bc(prob, data, u)

x0 = u(1:2);
x1 = u(3:4);
T0 = u(5);
T  = u(6);

y = [x1(1:2)-x0(1:2); T0; T-2*pi];

end
```

The corresponding restricted continuation problem is expressed in terms of 104 continuation variables, 104 constraints, and two inactive continuation parameters. We obtain a one-dimensional solution manifold by releasing `'p1'` as shown in the commands below.

```
>> prob = coco_set(prob, 'cont', 'atlas', '1d', 'PtMX', 200,  ...
     NAdapt', 0, 'h0', .2, 'h_max', .2, 'h_min', .2);
>> coco(prob, 'dim_1_atlas_1d', [], 1, 'p1', [0.2 2]);
```

Here, the call to the `coco` entry-point function is preceded by a selection of the `'atlas_1d'` atlas algorithm and an assignment of numerical values to the various atlas algorithm settings. Specifically, continuation here proceeds for up to 200 steps in either direction of continuation on the interval $p_1 \in [0.2, 2]$, with a fixed step size of 0.2 and without adaptive remeshing of the trajectory discretization. The resulting screen output is shown below.

```
    STEP   DAMPING               NORMS              COMPUTATION TIMES
  IT SIT     GAMMA     ||d||     ||f||     ||U||   F(x)  DF(x)  SOLVE
   0                           6.76e-03  9.61e+00    0.0    0.0    0.0
   1   1  1.00e+00  2.60e-03  2.26e-12  9.62e+00    0.0    0.0    0.0
   2   1  1.00e+00  4.26e-12  2.66e-15  9.62e+00    0.0    0.0    0.0

 STEP      TIME        ||U||  LABEL  TYPE           p1
    0  00:00:00   9.6166e+00      1  EP      1.0000e+00
   10  00:00:00   9.3679e+00      2          7.1548e-01
   20  00:00:00   8.7757e+00      3          3.8379e-01
   25  00:00:00   8.4289e+00      4  EP      2.0000e-01

 STEP      TIME        ||U||  LABEL  TYPE           p1
    0  00:00:00   9.6166e+00      5  EP      1.0000e+00
   10  00:00:00   9.4886e+00      6          1.2845e+00
```

```
  20  00:00:01   9.0522e+00      7          1.6162e+00
  29  00:00:01   8.5720e+00      8  EP      2.0000e+00
```

As a comparison, consider the following screen output, obtained after reconstructing the continuation problem with the 'NTST' option of the 'coll' toolbox set equal to 50.

```
    STEP   DAMPING              NORMS              COMPUTATION TIMES
 IT SIT     GAMMA     ||d||     ||f||     ||U||    F(x)   DF(x)   SOLVE
  0                           1.52e-02  1.71e+01    0.0    0.0    0.0
  1   1  1.00e+00  5.81e-03  8.28e-13  1.71e+01    0.0    0.0    0.0
  2   1  1.00e+00  2.16e-12  5.41e-15  1.71e+01    0.0    0.0    0.0

STEP      TIME        ||U||  LABEL  TYPE           p1
   0  00:00:00   1.7102e+01      1  EP      1.0000e+00
  10  00:00:00   1.6972e+01      2          8.7333e-01
  20  00:00:00   1.6615e+01      3          7.4257e-01
  30  00:00:00   1.6037e+01      4          6.0308e-01
  40  00:00:00   1.5253e+01      5          4.4902e-01
  50  00:00:01   1.4285e+01      6          2.7225e-01
  54  00:00:01   1.3892e+01      7  EP      2.0000e-01

STEP      TIME        ||U||  LABEL  TYPE           p1
   0  00:00:01   1.7102e+01      8  EP      1.0000e+00
  10  00:00:01   1.7002e+01      9          1.1267e+00
  20  00:00:01   1.6676e+01     10          1.2574e+00
  30  00:00:01   1.6136e+01     11          1.3969e+00
  40  00:00:02   1.5397e+01     12          1.5510e+00
  50  00:00:02   1.4488e+01     13          1.7277e+00
  60  00:00:02   1.3446e+01     14          1.9394e+00
  63  00:00:02   1.3171e+01     15  EP      2.0000e+00
```

In this case, the number of continuation variables equals 500 and this increase is reflected in the interpretation of the step size, resulting in an increase in the number of steps in each direction. A similar change occurs if we allow adaptive remeshing of the trajectory discretization by setting the 'NAdapt' option of the atlas algorithm to 1, as shown in the screen output below.

```
    STEP   DAMPING              NORMS              COMPUTATION TIMES
 IT SIT     GAMMA     ||d||     ||f||     ||U||    F(x)   DF(x)   SOLVE
  0                           1.52e-02  1.71e+01    0.0    0.0    0.0
  1   1  1.00e+00  5.81e-03  8.28e-13  1.71e+01    0.0    0.0    0.0
  2   1  1.00e+00  2.16e-12  5.41e-15  1.71e+01    0.0    0.0    0.0

STEP      TIME        ||U||  LABEL  TYPE           p1
   0  00:00:00   1.7102e+01      1  EP      1.0000e+00
  10  00:00:00   1.4784e+01      2          8.6116e-01
  20  00:00:00   1.2654e+01      3          6.9144e-01
  30  00:00:01   1.0139e+01      4          4.5936e-01
  39  00:00:01   9.4519e+00      5  EP      2.0000e-01

STEP      TIME        ||U||  LABEL  TYPE           p1
   0  00:00:01   1.7102e+01      6  EP      1.0000e+00
  10  00:00:01   1.4821e+01      7          1.1388e+00
  20  00:00:02   1.2752e+01      8          1.3086e+00
```

```
  30  00:00:02   1.0350e+01      9          1.5406e+00
  40  00:00:03   9.6533e+00     10          1.8653e+00
  44  00:00:03   9.2725e+00     11  EP      2.0000e+00
```

In this case, the number of continuation variables changes during continuation from a maximum of 500 to a minimum of 150.

As an alternative to `'atlas_1d'`, we demonstrate the application of `'atlas_kd'` to continuation along the same one-dimensional manifold. Since `'atlas_kd'` stores function data associated with each chart, we invoke the call

```
>> coco_func_data.pointers('set', []);
```

before problem construction in order to free up previously allocated memory. With `'NTST'` equal to 10, the commands

```
>> prob = coco_set(prob, 'cont', 'atlas', 'kd',  'PtMX', 200,  ...
  'NAdapt', 0, 'R', .2, 'R_max', .2, 'R_min', .2);
>> coco(prob, 'dim_1_atlas_kd', [], 1, 'p1', [0.2 2]);
```

then generate the screen output

```
    STEP   DAMPING               NORMS              COMPUTATION TIMES
 IT SIT     GAMMA     ||d||     ||f||     ||U||   F(x)  DF(x)  SOLVE
  0                           6.76e-03  9.61e+00    0.0    0.0    0.0
  1   1  1.00e+00  2.60e-03  2.26e-12  9.62e+00    0.0    0.0    0.0
  2   1  1.00e+00  4.26e-12  2.66e-15  9.62e+00    0.0    0.0    0.0

STEP      TIME        ||U||  LABEL  TYPE           p1
   0  00:00:00   9.6166e+00      1  EP      1.0000e+00
   9  00:00:00   8.4289e+00      2  EP      2.0000e-01
  10  00:00:00   8.6184e+00      3          1.9548e+00
  11  00:00:00   8.5720e+00      4  EP      2.0000e+00
```

In contrast to the use of `'atlas_1d'`, the output remains virtually the same when `'NTST'` is increased to 50, as shown below

```
    STEP   DAMPING               NORMS              COMPUTATION TIMES
 IT SIT     GAMMA     ||d||     ||f||     ||U||   F(x)  DF(x)  SOLVE
  0                           1.52e-02  1.71e+01    0.0    0.0    0.0
  1   1  1.00e+00  5.81e-03  8.28e-13  1.71e+01    0.0    0.0    0.0
  2   1  1.00e+00  2.16e-12  5.41e-15  1.71e+01    0.0    0.0    0.0

STEP      TIME        ||U||  LABEL  TYPE           p1
   0  00:00:00   1.7102e+01      1  EP      1.0000e+00
   9  00:00:00   1.3892e+01      2  EP      2.0000e-01
  10  00:00:01   1.3360e+01      3          1.9582e+00
  11  00:00:01   1.3171e+01      4  EP      2.0000e+00
```

In this case, if we let `'NAdapt'` equal 1, the screen output

```
    STEP   DAMPING               NORMS              COMPUTATION TIMES
 IT SIT     GAMMA     ||d||     ||f||     ||U||   F(x)  DF(x)  SOLVE
  0                           1.52e-02  1.71e+01    0.0    0.0    0.0
  1   1  1.00e+00  5.81e-03  8.28e-13  1.71e+01    0.0    0.0    0.0
```

```
   2   1  1.00e+00  2.16e-12  5.41e-15  1.71e+01    0.0    0.0    0.0

 STEP      TIME        ||U||  LABEL  TYPE           p1
    0  00:00:00   1.7102e+01      1  EP      1.0000e+00
    9  00:00:01   1.2985e+01      2  EP      2.0000e-01
   10  00:00:01   1.2556e+01      3          1.9581e+00
   11  00:00:02   1.2185e+01      4  EP      2.0000e+00
```

corresponds to variations in the number of continuation variables from a maximum of 500 to a minimum of 400.

We may use the `plot_atlas_kd` utility to visualize the result of continuation using the `'atlas_kd'` atlas algorithm. To this end, we append the call

```
>> prob = coco_add_pars(prob, 'vel', uidx(maps.x0_idx(2)), 'vel', 'active');
```

to the problem construction. The commands

```
>> prob = coco_set(prob, 'cont', 'atlas', 'kd',  'PtMX', 200,  ...
     'NAdapt', 1, 'R', .2, 'R_max', 2, 'R_min', .02);
>> coco(prob, 'dim_1_atlas_kd', [], 1, {'p1' 'vel'}, [0.2 2]);
```

modify the values of the `'R_min'` and `'R_max'` to ensure properly resolved curve segments during continuation. The commands

```
>> atlas = coco_bd_read('dim_1_atlas_kd', 'atlas');
>> figure; plot_atlas_kd(atlas.charts,2)
```

extract the atlas from the stored solution data and visualize the polyhedral tesselation of the solution manifold in terms of the projection onto the `'p1'` and `'vel'` continuation parameter space.

Finally, the commands

```
>> prob = coco_set(prob, 'cont', 'atlas', 'kd',  'PtMX', 1000, ...
     'NAdapt', 1, 'R', .1, 'R_max', 10, 'R_min', .01);
>> coco(prob, 'dim_2_atlas_kd', [], 2, {'p1' 'p2' 'vel'}, {[0.2 2], [0 1]});
```

result in continuation along the two-dimensional solution manifold obtained with both `'p1'` and `'p2'` allowed to vary. Visualization of the polyhedral tesselation projected onto the `'p1'`, `'p2'` and `'vel'` continuation paramter space results from the commands

```
>> atlas = coco_bd_read('dim_2_atlas_kd', 'atlas');
>> figure; plot_atlas_kd(atlas.charts,3)
```

The command

```
>> plot_atlas_kd(atlas.charts,3,'basepoints')
```

overlays the locus of base points of individual charts.

---

**Exercises**

1. Repeat the analysis in this section using an autonomous implementation of the governing vector field and with explicit Jacobians.

2. Repeat the construction of a two-dimensional solution manifold using `'atlas_kd'` by projecting onto `'p1'`, `'p2'` and an active continuation parameter corresponding to the maximum velocity along the periodic orbit.

3. Explore the various optional settings of the `'atlas_1d'` and `'atlas_kd'` atlas algorithms.

4. Revisit the atlas algorithms in Part III of *Recipes for Continuation* and investigate a modification that supports global comparisons of solutions using a subset of continuation variables that are invariant in number and interpretation.

---