> help bvp5c

bvp5c Solve boundary value problems for ODEs by collocation.

SOL = bvp5c(ODEFUN,BCFUN,SOLINIT) integrates a system of ordinary

differential equations of the form y' = f(x,y) on the interval [a,b],

subject to general two-point boundary conditions of the form

bc(y(a),y(b)) = 0. ODEFUN and BCFUN are function handles. For a scalar X

and a column vector Y, ODEFUN(X,Y) must return a column vector representing

f(x,y). For column vectors YA and YB, BCFUN(YA,YB) must return a column

vector representing bc(y(a),y(b)). SOLINIT is a structure with fields

x -- ordered nodes of the initial mesh with

SOLINIT.x(1) = a, SOLINIT.x(end) = b

y -- initial guess for the solution with SOLINIT.y(:,i)

a guess for y(x(i)), the solution at the node SOLINIT.x(i)

bvp5c produces a solution that is continuous on [a,b] and has a

continuous first derivative there. The solution is evaluated at points

XINT using the output SOL of bvp5c and the function DEVAL:

YINT = DEVAL(SOL,XINT). The output SOL is a structure with

SOL.solver -- 'bvp5c'

SOL.x -- mesh selected by bvp5c

SOL.y -- approximation to y(x) at the mesh points of SOL.x

SOL.stats -- computational cost statistics (also displayed when

the 'Stats' option is set with BVPSET).

SOL = bvp5c(ODEFUN,BCFUN,SOLINIT,OPTIONS) solves as above with default

parameters replaced by values in OPTIONS, a structure created with the

BVPSET function. To reduce the run time greatly, use OPTIONS to supply

a function for evaluating the Jacobian and/or vectorize ODEFUN.

See BVPSET for details and SHOCKBVP for an example that does both.

Some boundary value problems involve a vector of unknown parameters p

that must be computed along with y(x):

y' = f(x,y,p)

0 = bc(y(a),y(b),p)

For such problems the field SOLINIT.parameters is used to provide a guess

for the unknown parameters. On output the parameters found are returned

in the field SOL.parameters. The solution SOL of a problem with one set

of parameter values can be used as SOLINIT for another set. Difficult BVPs

may be solved by continuation: start with parameter values for which you can

get a solution, and use it as a guess for the solution of a problem with

parameters closer to the ones you want. Repeat until you solve the BVP

for the parameters you want.

The function BVPINIT forms the guess structure in the most common

situations: SOLINIT = BVPINIT(X,YINIT) forms the guess for an initial

mesh X as described for SOLINIT.x, and YINIT either a constant vector

guess for the solution or a function handle. If YINIT is a function handle

then for a scalar X, YINIT(X) must return a column vector, a guess for

the solution at point x in [a,b]. If the problem involves unknown parameters

SOLINIT = BVPINIT(X,YINIT,PARAMS) forms the guess with the vector PARAMS of

guesses for the unknown parameters.

bvp5c solves a class of singular BVPs, including problems with

unknown parameters p, of the form

y' = S\*y/x + f(x,y,p)

0 = bc(y(0),y(b),p)

The interval is required to be [0, b] with b > 0.

Often such problems arise when computing a smooth solution of

ODEs that result from PDEs because of cylindrical or spherical

symmetry. For singular problems the (constant) matrix S is

specified as the value of the 'SingularTerm' option of BVPSET,

and ODEFUN evaluates only f(x,y,p). The boundary conditions

must be consistent with the necessary condition S\*y(0) = 0 and

the initial guess should satisfy this condition.

bvp5c can solve multipoint boundary value problems. For such problems

there are boundary conditions at points in [a,b]. Generally these points

represent interfaces and provide a natural division of [a,b] into regions.

bvp5c enumerates the regions from left to right (from a to b), with indices

starting from 1. In region k, bvp5c evaluates the derivative as

YP = ODEFUN(X,Y,K). In the boundary conditions function,

BCFUN(YLEFT,YRIGHT), YLEFT(:,K) is the solution at the 'left' boundary

of region k and similarly for YRIGHT(:,K). When an initial guess is

created with BVPINIT(XINIT,YINIT), XINIT must have double entries for

each interface point. If YINIT is a function handle, BVPINIT calls

Y = YINIT(X,K) to get an initial guess for the solution at X in region k.

In the solution structure SOL returned by bvp5c, SOL.x has double entries

for each interface point. The corresponding columns of SOL.y contain

the 'left' and 'right' solution at the interface, respectively.

See THREEBVP for an example of solving a three-point BVP.

Example

solinit = bvpinit([0 1 2 3 4],[1 0]);

sol = bvp5c(@twoode,@twobc,solinit);

solve a BVP on the interval [0,4] with differential equations and

boundary conditions computed by functions twoode and twobc, respectively.

This example uses [0 1 2 3 4] as an initial mesh, and [1 0] as an initial

approximation of the solution components at the mesh points.

xint = linspace(0,4);

yint = deval(sol,xint);

evaluate the solution at 100 equally spaced points in [0 4]. The first

component of the solution is then plotted with

plot(xint,yint(1,:));

For more examples see FSBVP, SHOCKBVP, MAT4BVP, EMDENBVP. To use the

bvp5c solver, you must pass 'bvp5c' as input argument:

fsbvp('bvp5c')

bvp5c is used exactly like BVP4C, but error tolerances do not mean the

same in the two solvers. If S(x) approximates the solution y(x), BVP4C

controls the residual |S'(x) - f(x,S(x))|. This controls indirectly the

true error |y(x) - S(x)|. bvp5c controls the true error directly.

bvp5c is more efficient than BVP4C for small error tolerances.

See also bvp4c, bvpset, bvpget, bvpinit, bvpxtend, deval, function\_handle.