INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique, d'Hydraulique et de Télécommunications

WASP: a Wavelet Adaptive Solver for boundary value Problems

Short Reference Manual

J. B. Caillau and J. Noailles

ENSEEIHT–IRIT, UMR CNRS 5505
Parallel Algorithms and Optimization group (APO)
2 rue Camichel, 31071 Toulouse, France

RT/APO/01/1

WASP: A WAVELET ADAPTIVE SOLVER FOR BOUNDARY VALUE PROBLEMS

SHORT REFERENCE MANUAL*

J. B. CAILLAU† AND J. NOAILLES†

Abstract. This is a short guide to use the *Matlab* package `WASP` designed for the numerical solution of two–point linear boundary value problems that arise typically in linear quadratic optimal control. The method relies upon an adaptive computation of discretization based on a wavelet analysis. On a given refined grid, finite differences of various order are used.

Key words. two–point linear boundary value problems, finite differences, adaptive discretization, grid refinement, wavelet analysis

AMS subject classifications. 65L10, 65L50, 42C40

1. Introduction. `WASP` is a small *Matlab* package designed for the numerical solution of two–point linear boundary value problems, $(LBVP)$, of the form

$$\dot{y} = A(t)y + b(t),\ t \in [t_0, t_f] \tag{1.1}$$

$$C_0 y(t_0) + C_f y(t_f) = d \tag{1.2}$$

where A and b are smooth functions, $A(t)$, C_0 and C_f being n by n matrices, $b(t)$ and d vectors in $\mathbf{R}^n$ (n is the dimension of the problem). This kind of problem originates for instance from the use of the first order necessary condition—the Pontryagin maximum principle—on linear quadratic optimal control problems (the so–called LQ Regulator). It can be approximated by a linear system using finite differences schemes [1]. The idea here is to do so, but using moreover a grid refinement process so as to choose adaptively the instants for the finite differences. Roughly speaking, thanks to a wavelet analysis performed on the approximation of the solution to (1.1)–(1.2) on a coarse grid (grid of resolution J), a new refined grid (of resolution $J+1$) is computed by adding points whenever high wavelet coefficients are detected. We refer to [3] for the details. The package requires *Matlab* 4 or higher, together with the wavelet library *WaveLab* [2], version v.701 or higher.

The paper is organized as follows. In section §2 the installation procedure is discussed, starting with the files retrieval on the Web; in section §3, two examples extracted from [3] are treated; section §4 gives the alphabetic function synopses and section §5 finally provides a review of each function of the package (*Matlab* help–like).

2. Access and installation. All files are downloadable at the following Web address (URL):

`www.enseeiht.fr/apo/wasp`

Since this is a *Matlab* package, any system on which *Matlab* (version 4 and higher) runs and for which the library *WaveLab* (version v.701 and higher) is available is supported. Hence the user has merely three steps to perform:

---

*This work was supported in part by the French Ministry for Education, Higher Education and Research (grant 20INP96) and by the French Space Agency (contracts 871/94/CNES/1454 and 86/776/98/CNES/7462).

†ENSEEIHT–IRIT, UMR CNRS 5505, 2 rue Camichel, 31071 Toulouse, France (`caillau@enseeiht.fr`, `jnoaille@enseeiht.fr`).

1. get the `WASP` archive containing the package at the indicated Web site;
2. have the library *WaveLab* installed;
3. indicate to *Matlab* the current path of the `M-files` contained in the `WASP` archive (file `WaspPath.m`, see below).

The library *WaveLab* [2] is accessible through the World Wide Web at:

`www-stat.stanford.edu/~wavelab`

Two formats for the `WASP` archive are available:

- gziped tar, file `wasp-v1.tar.gz`;
- ziped file `wasp-v1.zip`.

Once unarchived, the directory `wasp-v1` must contain the following files (check with `readme` file):

```
wasp-v1:
demo        readme      source      startup.m   WaspPath.m

wasp-v1/demo:
Afeps.m   bseps.m   d2seps.m  dseps.m   seps.m    test2.m
Aseps.m   d2feps.m  dfeps.m   feps.m    sinv.m
bfeps.m   d2psi.m   dpsi.m    psi.m     test1.m

wasp-v1/source:
disc.m      gup.m       PlotDisc.m  rksolve.m   top.m
getg.m      lc2f.m      rk.m        tip.m       wasp.m
```

The last step is to update the file `WaspPath.m` by indicating the right path (that is the absolute path of the directory `wasp-v1` you have unarchived). This is done by updating properly the line:

```
WASPPATH = '/homes/caillau/rec/code/wasp/wasp-v1/';
```

3. Two examples. Two examples, borrowed from [3], are given as demos. To run them, copy the files `WaspPath.m` and `startup.m` in a directory (any); of course, you also need to copy the file `WavePath.m` provided by *WaveLab* in the same directory (both `WavePath.m` and `WaspPath.m` are called by `startup.m`). Then, run *Matlab* and use the commands `test1` or `test2`. You shall get the figures 3.1 and 3.2 as well as the messages below (depending on your system, the execution times may vary; these have been obtained on a *Sparc* 20):

```
>> test1
RKSOLVE infos (Gauss2)
Condition number for N =   32 : 279.509519
Total time                    : 1.090000
Assembly                      : 98.2 %
Factorization                 : 1.8 %
RKSOLVE infos (Gauss2)
Condition number for N =   35 : 312.419021
Total time                    : 1.050000
Assembly                      : 99.0 %
Factorization                 : 1.0 %
RKSOLVE infos (Gauss2)
Condition number for N =   48 : 418.324986
Total time                    : 1.480000
Assembly                      : 98.0 %
Factorization                 : 2.0 %
RKSOLVE infos (Gauss2)
Condition number for N =   72 : 630.622307
```

```
Total time                  : 2.200000
Assembly                    : 97.7 %
Factorization               : 2.3 %
RKSOLVE infos (Gauss2)
Condition number for N =  113 : 966.629198
Total time                  : 3.510000
Assembly                    : 97.2 %
Factorization               : 2.8 %
RKSOLVE infos (Gauss2)
Condition number for N =  183 : 1513.843936
Total time                  : 5.830000
Assembly                    : 95.7 %
Factorization               : 4.3 %
t = 21.640000   sl = 73.7 %   disc = 26.3 %
Check Ns, cds, tsls and afs


>> test2
RKSOLVE infos (Gauss3)
Condition number for N =  512 : 2475.860160
Total time                  : 22.000000
Assembly                    : 90.5 %
Factorization               : 9.5 %
RKSOLVE infos (Gauss3)
Condition number for N =  557 : 2641.464924
Total time                  : 24.340000
Assembly                    : 89.8 %
Factorization               : 10.2 %
RKSOLVE infos (Gauss3)
Condition number for N =  661 : 2996.424233
Total time                  : 29.310000
Assembly                    : 88.0 %
Factorization               : 12.0 %
RKSOLVE infos (Gauss3)
Condition number for N =  875 : 3720.312805
Total time                  : 40.940000
Assembly                    : 84.7 %
Factorization               : 15.3 %
RKSOLVE infos (Gauss3)
Condition number for N = 1344 : 5286.139055
Total time                  : 69.230000
Assembly                    : 78.3 %
Factorization               : 21.7 %
RKSOLVE infos (Gauss3)
Condition number for N = 2215 : 8165.863567
Total time                  : 132.640000
Assembly                    : 69.6 %
Factorization               : 30.4 %
RKSOLVE infos (Gauss3)
Condition number for N = 3961 : 13895.340808
Total time                  : 312.760000
Assembly                    : 57.4 %
Factorization               : 42.6 %
RKSOLVE infos (Gauss3)
```

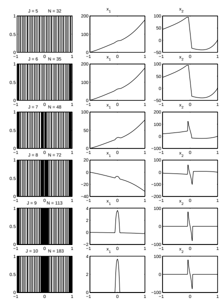

FIG. 3.1. *Figure (zoomable) produced by test1. Compare fig. 1, left, in [3].*

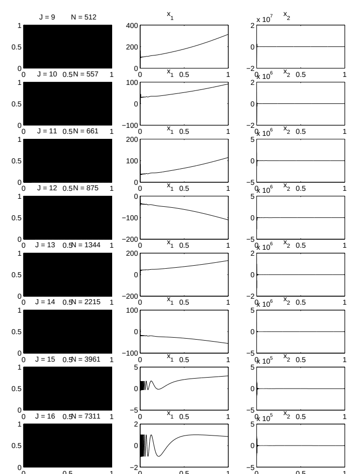

FIG. 3.2. *Figure (zoomable) produced by test2. Compare fig. 1, right, in [3]. Here, the grid—and not the density graph of the points—is plotted; let's recognize it, beyond a thousand points, it's getting hard to see something without zooming!*

```
Condition number for N = 7311 : 24910.668402
Total time                    : 828.600000
Assembly                      : 45.6 %
Factorization                 : 54.4 %
t = 1974.910000   sl = 74.4 %   disc = 25.6 %
Check Ns, cds, tsls and afs
```

The two files `test1.m` and `test2.m` can be used as examples of typical drivers for calls to the function `wasp` (see §5).

4. Alphabetic synopses of functions.

```
[t2,tl2] = disc(t,X,tl,etha,t0,tf,J,JMAX,Type,Par)
t = getg(tl,t0,tf)
tl2 = gup(tl,w,etha,J,JMAX)
y = lc2f(t,x,N)
n = PlotDisc(base,t,d)
[Alpha,beta,gamma] = rk(RK)
[X,cd,af] = rksolve(Afun,bfun,C0,Cf,d,t,RK)
counter = tip
elapsed = top(counter)
[t,X,Ns,cds,tsls,afs] = ...
wasp(Afun,bfun,t0,tf,C0,Cf,d,RK,JMIN,JMAX,Type,Par,etha)
```

5. Review of WASP v.1.

disc

```
function [t2,tl2] = disc(t,X,tl,etha,t0,tf,J,JMAX,Type,Par)
 disc -- Grid updating

  Usage
    [t2,tl2] = disc(t,X,tl,etha,t0,tf,J,JMAX,Type,Par)

  Inputs
    t      physical grid
    X      signal on t
    tl     logical grid associated with t
    etha   ratio of coeffs to keep on W_J
    t0     initial time
    tf     final time
    J      current resolution
    JMAX   maximum resolution
    Type   type of wavelet (see MakeONFilter)
    Par    parameter of wavelet (idem)

  Outputs
    t2     physical grid updated
    tl2    logical grid updated

  Description
    Refines the physical and logical grids on which is defined
    the signal x thanks to a Wavelet Analysis of each dimension.

  See also
    lc2f, MakeONFilter, FWT_PO, gup, getg

  References
    WaveLab
```

getg

```
function t = getg(tl,t0,tf)
 getg -- Gets grid

  Usage
    t = getg(tl,t0,tf)

  Inputs
    tl     logical grid
    t0     initial time
    tf     final time

  Outputs
    t      physical grid

  Description
    Computes the physical grid associated with a logical one.

  See also
    rksolve
```

gup

```
function tl2 = gup(tl,w,etha,J,JMAX)
 gup -- Grid updating

  Usage
    tl = gup(tl,w,etha,J,JMAX)

  Inputs
    tl      logical grid
    w       wavelet coefficients on W_J
    etha    ratio to keep
    J       current resolution
    JMAX    maximum resolution

  Outputs
    tl2      physical grid updated

  Description
    Updates the logical grid tl, using wavelets coeffs on W_J,
    from resolution J to resolution J+1.

  See also
    rksolve, FWT_PO, KeepBiggest
```

lc2f

```
function y = lc2f(t,x,N)
 lc2f -- linear coarse to fine

  Usage
    y = lc2f(t,x,N)

  Inputs
    t       coarse grid
    x       data on coarse grid
    N       size of the regular fine grid

  Outputs
    y       linearly interpolated data on the fine grid

  Description
    Linear interpolation of the data x, given on the coarse
    grid t, on the fine regular grid of size N.

  See also
    sample
```

PlotDisc

```
function n = PlotDisc(base,t,d)
 PlotDisc -- Discretization plot

  Usage
    n = PlotDisc(base,t,d)

  Inputs
    base   ref value
    t      discretization grid
    d      max deviation from ref

  Outputs
    n      grid size

  Description
    Plot the associated grid (spike-like).

  See also
    PlotSpikes
```

rk

```
function [Alpha,beta,gamma] = rk(RK)
 rk -- Runge-Kutta scheme

  Usage
    [Alpha,beta,gamma] = rk(RK)

  Inputs
    RK      string, 'Gauss1', 'Gauss2', 'Gauss3',
                    'Lobatto2', 'Lobatto3',
                    'Radau1',
                    'Exp4', 'Exp5'

  Outputs
    Alpha   quadrature matrix
    beta    quadrature vector
    gamma   collocation points

  Description
    Alpha, beta and gamma define the coefficients to
    compute :

             y := x + h . phi(t,x,h)
    phi(t,x,h) := sum(k=1,r) beta(k) . f(tk,xk)
            tk := t + gamma(k) . h
            xk := x + h . sum(l=1,r) Alpha(k,l) . f(tl,xl)

  See also
    rksolve

  References
    "Analyse numerique des equations differentielles",
    M. Crouzeix, L. Mignot,
    "Numerical solution of boundary value problems for
    ordinary differential equations", U. M. Ascher,
    R. M. M. Mattheij, R. D. Russel.
```

rksolve

```
function [X,cd,af] = rksolve(Afun,bfun,C0,Cf,d,t,RK)
 rksolve -- Solution of (LBVP) by Runge-Kutta

  Usage
    [X,cd,af] = rksolve(Afun,bfun,C0,Cf,d,t,RK)

  Inputs
    Afun   function defining A(t)
    bfun   function defining b(t)
    C0     boundary conditions
    Cf     idem
    d      idem
    t      discretization grid
    RK     string, type of Runge-Kutta used (see rk)

  Outputs
    X      solution on the grid t
    cd     condition number of the linear system
    af     factorization time (vs assembling) ratio

  Description
    Solution of

    (LBVP) dx/dt = A(t).x + b(t) , t in [t0,tf]
           C0.x(t0) + Cf.x(tf) = d

    using Runge-Kutta.

  See also
    rk

  References
    "Numerical solution of boundary value problems for
    ordinary differential equations", U. M. Ascher,
    R. M. M. Mattheij, R. D. Russel.
```

tip

```
function counter = tip
 tip -- starts a new counter

  Usage
    counter = tip

  Outputs
    counter counter handler

  Description
    Starts a new cputime counter.

  See also
    top, cputime
```

top

```
function elapsed = top(counter)
 elapsed -- elapsed CPU time

  Usage
    elapsed = top(counter)

  Inputs
    counter counter handler (optional)

  Outputs
    elpased elapsed CPU time for counter

  Description
    Elapsed CPU time since counter was created by tip.
    If no handler is specified, takes the most recent
    one.

  See also
    tip, cputime
```

wasp

```
function [t,X,Ns,cds,tsls,afs] = ...
wasp(Afun,bfun,t0,tf,C0,Cf,d,RK,JMIN,JMAX,Type,Par,etha)
 wasp -- Wavelet Adaptive Solver for boundary value Problems

  Usage
    [t,X,Ns,cds,tsls,afs] = ...
    wasp(Afun,bfun,t0,tf,C0,Cf,d,RK,JMIN,JMAX,Type,Par,etha)

  Inputs
    Afun   function defining A(t)
    bfun   function defining b(t)
    t0     initial time t0
    tf     final time tf
    C0     boundary conditions
    Cf     idem
    d      idem
    RK     string, type of Runge-Kutta used (see rk)
    JMIN   minimum resolution
    JMAX   maximum resolution
    Type   type of wavelet (see MakeONFilter)
    Par    parameter of wavelet (idem)
    etha   compression ratio

  Outputs
    t      adapted grid
    X      solution on the adapted grid
    Ns     grid sizes
    tsls   rksolve's elapsed times
    cds    condition numbers (LU factorizations)
    afs    factorization ratios (factorization versus assembling)

  Description
    Solution of

    (LBVP) dx/dt = A(t).x + b(t) , t in [t0,tf]
           C0.x(t0) + Cf.x(tf) = d

    using an adaptive Runge-Kutta like scheme. The adaptive
    discretization is computed using a wavelet analysis. The
    supporting library used for wavelets is WaveLab v.701 (c).

  See also
    rksolve, disc

  References
    "Wavelets for adaptive solution of boundary value problems",
    J. B. Caillau and J. Noailles, Proceedings of the 16th IMACS
    Conference, August 2000, Lausanne, Switzerland.
```

REFERENCES

[1] U. M. Ascher, R. M. M. Mattheij, and R. D. Russel, *Numerical solution of boundary value problems for differential equations*, Prentice Hall, 1988.

[2] J. Buckheit, S. Chen, D. Donoho, I. Johnstone, and J. Scargle, *WaveLab reference manual*, tech. report, Stanford University, 1995.

[3] J. B. Caillau and J. Noailles, *Wavelets for adaptive solution of boundary value problems*, in Proceedings of the 16th IMACS Conference, M. Deville and R. Owens Eds., Lausanne, Switzerland, August 2000.