4.9 A Summary

Once rendered discrete for placement on a digital computer, time-dependent inverse problems all reduce formally to the least-squares problems already considered in Chapters 2, 3. With a large enough, fast enough computer, they could even be solved by the same methods used for the static problems. Given however, the common need to to reduce the computing and storage burdens, a number of algorithms are available for finding solutions by considering the problem either in pieces, as in the sequential methods of filter/smoother, or by iterations as in the Lagrange multiplier methods. Solutions can be either accurate or approximate depending upon one's needs and resources. In the end, however, the main message is that one is still seeking the solution to a least-squares problem, and the differences among the techniques are algorithmic ones, with tradeoffs of convenience and cost.

Appendix to Chapter. Automatic Differentiation and Adjoints

The utility of automatic differentiation (AD) of computer model codes was alluded to on Pages, 4.2.2, 4.7.3, both as a way to determine the state transition matrix $\mathbf{A}$, when it was only implicit in a code, and as a route to linearizing nonlinear models. The construction of software capable of taking (say) a Fortran90 code and automatically generating a second Fortran90 code for the requisite derivatives of the model is a considerable, if not altogether complete, achievement of computer science. Any serious discussion is beyond the author's expertise, and well outside the scope of this book. But because only AD methods have made the Lagrange multiplier (adjoint) method of state estimation a practical approach for realistic fluid problems, we briefly sketch the possibilities with a few simple examples. The references given in note 142 should be consulted for a proper discussion.

Consider first the problem of finding the state transition matrix. A simple time stepping code written in Matlab for a 2-vector is

```
function y=lin(x);
y(1)=0.9*x(1)+0.2*x(2); y(2)=0.2*x(1)+0.8*x(2);
```

Here $\mathbf{x}$ would be the state vector at time $t-1$, and $\mathbf{y}$ would be its value one-time step in the future. A matrix/vector notation is deliberately avoided so that $\mathbf{A}$ is not explicitly specified. When the AD tool ADiMat (available through http://www.sc.rwth-aachen.de/adimat) is used, it writes a new Matlab code,

```
function [g_y, y]= g_lin(g_x, x) %lin.m;
```

```
 %x is assumed to be a 2-vector
g_lin_0= 0.9* g_x(1);
 lin_0= 0.9* x(1);
 g_lin_1= 0.2* g_x(2);
 lin_1= 0.2* x(2);
 g_y(1)= g_lin_0+ g_lin_1;
 y(1)= lin_0+ lin_1;
 clear lin_0 lin_1 g_lin_0 g_lin_1 ;
 g_lin_2= 0.2* g_x(1);
 lin_2= 0.2* x(1);
 g_lin_3= 0.8* g_x(2);
 lin_3= 0.8* x(2);
 g_y(2)= g_lin_2+ g_lin_3;
 y(2)= lin_2+ lin_3;
 clear lin_2   lin_3   g_lin_2   g_lin_3 ;
```

The notation has been cleaned up somewhat to make it more readable. Consider for example, the new variable, `g_lin_0` $= 0.9 *$ `g_x(1)`; `The` numerical value 0.9 is the partial derivative of $y\,(1)$ with respect to $x\,(1)$. The variable `g_x(1)` would be the partial derivative of $x\,(1)$ with respect to some other independent variable, permitting the chain rule to operate if desired. Otherwise, one can set it to unity on input. Similarly the notation `g_lin_i` denotes the corresponding derivative of $y(1)$ with respect to $x(i)$. By simple further coding, one can construct the **A** matrix of the values of the partial derivatives. Here, ADiMat has produced the tangent linear model, which is also the exact forward model. More interesting examples can be constructed.

The Matlab code corresponding to the simple switch is,

```
function y= switch1(a); if a> 0, y= a; else, y= a^2+2*a; end
```

that is, $y = a$ if independent variable a (externally prescribed) is positive, or else, $y = a^2 + 2a$. Running this code through AdiMat produces (again after some cleaning up of the notation),

```
function [g_y, y]= g_switch1(g_a, a);
 if a> 0, g_y= g_a;
 y= a;
else, g_tmp_2=2* a^(2- 1)* g_a;
 tmp_0= a^2;
 g_tmp_1= 2* g_a;
 tmp_1= 2* a;
 g_y= g_tmp_0+ g_tmp_1;
```

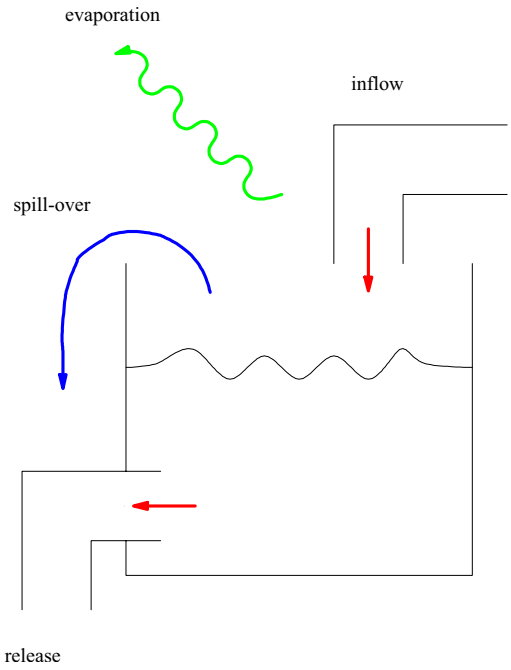

Figure 4.18: Reservoir. (From P. Heimbach) {reservoir.eps

```
y= tmp_0+ tmp_1;
end
```

That is to say, the derivative, `g_y`, is 1 for positive a, otherwise it is $a + 2$. `g_a` can be interpreted as the derivative of a with respect to another, arbitrary, independent variable, thus permitting the use of the chain rule. The derivative is defined for all values of a except $a = 0$.

Consider now a physical problem of a reservoir as shown in Fig. 4.18.[151] The model is chosen specifically to have discontinuous behavior: there is an inflow, storage, and outflow. If the storage capacity is exceeded, there can be overspill, determined by the `max` statement below. A forward code, now written in Fortran is,

Thresholds: a hydrological reservoir model (I)

```
do t = 1,  msteps
```

- get sources & sinks at time t
  inflow, evaporation, release
- calculate water release based on storage
  `release(t) = 0.8*storage(t-1)`$^{0.7}$
- calculate projected stored water,

storage = storage+inflow-release-evaporation

```
nominal = storage(t-1) +
   h*( infl(t)-release(t)-evap(t) )
```

- If threshold capacity is exceeded, spill-over:
  `spill(t) = MAX(nominal-capac , 0.)`
- re-adjust projected stored water after spill-over:
  `storage(t) = nominal - spill(t)`
- determine outflow:
  `out(t) = release(t) + spill(t)/h`

`end do`

Note the presence of the `max` statement.

When run through the AD tool TAF (a product of FastOpt®), one obtains for the tangent linear model,

Thresholds: a hydrological reservoir model (II)

The tangent linear model

```
do t=1, msteps
g_release(t) =0.56*g_storage(t-1)*storage(t-1)**(-0.3)
release(t)=0.8*storage(t-1)**0.7
g_nominal=-g_release(t)*h+g_storage(t-1)
nominal=storage(t-1)+h*(infl(t)-release(t)-evap(t))
g_spill(t)=g_nominal*(0.5+sign(0.5,nominal-capac-0.))
spill(t)=MAX(nominal-capac,0.)
g_storage(t)=g_nominal-g_spill(t)
storage(t)=nominal-spill(t)
g_out(t)=g_release(t)+g_spill(t)/h
out(t)=release(t)+spill(t)/h
end do
```

- `g_release(t)` not defined for `storage(t-1) = 0.`
- `g_spill(t)` not defined for `nominal = capac`

Note how the maximum statement has given rise to the new variable `g_spill(t)`, its corresponding adjoint variable.

Note that the AD tool can cope with such apparently non-differentiable operators as the maximum of a vector. In practice, it internally replaces the function `max` with a loop of tests for relative sizes of successive elements. Not all AD tools can cope with all language syntaxes, and one must be alert to failures owing to incomplete handling of various structures. Nonetheless, the existing tools are a remarkably complete achievement.

TAF is capable of producing the reverse mode. A major issue is the ability to restart the computation from intermediate results, in an operation called "checkpointing."

Notes

[101]Liebelt (1967), Gelb (1974), Bryson and Ho (1975), Brown (1983), and Anderson and Moore (1979) are especially helpful

[102]Daley (1991)

[103]Meteorologists have tended to go their own idiosynchratic way—seee Ide et al. (1997)—with some loss in transparency to other fields.

[104]Box et al. (1994)

[105]Luenberger (1979)

[106]Menemenlis and Wunsch (1997); Stammer and Wunsch (1996)

[107]von Storch, et al. (1988).

[108]Giering and Kaminski (1997), Marotzke et al. (1999).

[109]See Bryson and Ho (1975, p. 351).

[110]For example, Stengel (1986).

[111]Munk, Worcester, & Wunsch (1995).

[112]A method exploited by Stammer and Wunsch (1996).

[113]Kalman (1960). Kalman's derivation was for this discrete case. The continuous case, which was derived later, is known as the "Kalman-Bucy" filter and is a much more complicated object.

[114]Stengel (1986, Eq. 4.3-22)

[115]For example, Goodwin and Sin (1984, p. 59).

[116]Feller (1957).

[117]Anderson and Moore (1979) discuss these and other variants of the Kalman filter equations.

[118]Some history of the idea of the filter, its origins in the work of Wiener and Kolmogoroff and with a number of applications, can be found in Sorenson (1985).

[119]Bryson & Ho (1975 , p. 363) , or Brown (1983, p. 218).

[120]Adapted from Bryson & Ho (1975, Chapter 13), whose notation is unfortunately somewhat difficult.

[121]For Rauch, Tung, and Striebel (1965)

[122]Gelb (1974); Bryson & Ho (1975); Anderson & Moore (1979); Goodwin & Sin, (1984); Sorenson (1985).

[123]Some guidance is provided by Bryson and Ho (1975, pp. 390–5) or Liebelt (1967). In particular, Bryson and Ho (1975) introduce the Lagrange multipliers (their equations 13.2.7–13.2.8) simply as an intermediate numerical device for solving the smoother equations.

[124]Luenberger (1979)

[125]Wunsch (1988b) shows a variety of calculations as a function of variations in the terminal constraint accuracies. An example of the used of this type of model is discussed in Chapter 6.

[126]Bennett (2002).

[127]The use of the adjoint to solve l_2-norm problems is discussed by Bryson and Ho (1975, Section 13.3), who relax the restriction of full controllability, $\mathbf{\Gamma} = \mathbf{I}$. Because of the connection to regulator/control problems, a variety of methods for solution is explored there.

[128]Bryson & Ho (1975)

[129]Franklin, Powell, and Workman (1990)

[130]See the examples in Ghil et al. (1981) or Figure 6–5e.

[131]Stengel (1986), Franklin (1990), Anderson and Moore (1979), Bittanti, Laub, and Willems (1991a), Fukumori et al. (1993) and Fu et al. (1993).

[132]See Reid (1972) for a discussion of the history of the Riccati equation in general; it is intimately related to Bessel's equation and has been studied in scalar form since the 18th century. Bittanti et al. (1991a) discuss many different aspects of the matrix form.

[133]Discussed by Bittanti et al. (1991b)

[134]following Franklin et al., 1990

[135]e.g., Franklin et al., 1990; Stengel, 1986

[136]Goodwin and Sin (1984) or Stengel (1986)

[137]Miller, Ghil, and Gauthiez (1994) discuss some of the practical difficulties.

[138]For example, Miller et al. (1994)

[139]e.g., Lea et al. (2000).

[140]E.g., Anderson & Moore, 1979; Goodwin & Sin, 1984; Haykin, 1986

[141]Among textbooks that discuss this subject are those of Haykin (1986), Goodwin and Sin (1984), and Ljung (1987).

[142]E.g., see Stengel, Chapter 5

[143]Giering and Kaminski (1998); Marotzke et al. (1990); Griewank (2000); Corliss et al. (2002)

[144]See Marotzke et al. (1999) or Giering (2000).

[145]Rall (1981), Griewank (2002)

[146]We follow here, primarily, Marotzke et al. (1999).

[147]Restrepo et al. (1995) discuss some of the considerations.

[148]e.g., Gill et al., 1981; Luenberger, 1984; Scales, 1985

[149]Gelb, 1974

[150]Luenberger, 1964; O'Reilly, 1983

[151]Example due to P. Heimbach. For more information see http://mitgcm.org, Short Course on Data Assimilation, WHOI, May 2003.