Source-to-source algorithmic differentiation

Ian Fenty
ECCO Summer School
May 2019

MITgcm Model Code Snippet

Two-step Adams–Bashforth calculation of RHS

Consider this arbitrary linear operation found in the MITgcm, calculating the RHS of the AB2 equation:

$$y_{n+2} = y_{n+1} + \frac{3}{2}hf(t_{n+1}, y_{n+1}) - \frac{1}{2}hf(t_n, y_n).$$

Forward

```
 subroutine adams_bashforth2( bi, bj, karg, ksize, gtracer, gtrnm1,
$ ab_gtr, startab, myiter, mythid )
```

Adjoint

```
 subroutine adams_bashforth2_ad( karg, ksize, gtracer_ad,
$gtrnm1_ad, ab_gtr_ad, startab, myiter )
```

MITgcm Model Code Snippet

Two-step Adams–Bashforth calculation of RHS

Consider this arbitrary linear operation found in the MITgcm, calculating the RHS of the AB2 equation:

$$y_{n+2} = y_{n+1} + \frac{3}{2}hf(t_{n+1}, y_{n+1}) - \frac{1}{2}hf(t_n, y_n).$$

Found in the MITgcm forward routine:

```
subroutine adams_bashforth2( bi, bj, karg, ksize, gtracer, gtrnm1,
$ ab_gtr, startab, myiter, mythid )
```

MITgcm Model Code Snippet

Two-step Adams–Bashforth calculation of RHS

Consider this arbitrary linear operation found in the MITgcm, calculating the RHS of the AB2 equation:

$$y_{n+2} = y_{n+1} + \frac{3}{2} h f(t_{n+1}, y_{n+1}) - \frac{1}{2} h f(t_n, y_n)$$

Forward model snippet

The MITgcm calculates in three lines of Fortran 77 code.

```
do j = 1-oly, sny+oly
  do i = 1-olx, snx+olx
    ab_gtr(i,j) = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
    gtrnm1(i,j,k) = gtracer(i,j,k)
    gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
  end do
end do
```

Forward and Adjoint Code

TAF processes the forward model source code, line by line, and constructs a new source code following a set of transformation algorithms. Here, the code on the left is transformed to the code on the right. Forward model variable names and their corresponding adjoint variables are colored.

Forward model code snippet

```
do j = 1-oly, sny+oly
  do i = 1-olx, snx+olx
    ab_gtr(i,j) = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
    gtrnm1(i,j,k) = gtracer(i,j,k)
    gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
  end do
end do
```

Adjoint code equivalent generated by TAF

```
do j = 1-oly, sny+oly
  do i = 1-olx, snx+olx
    ab_gtr_ad(i,j) = ab_gtr_ad(i,j) +gtracer_ad(i,j,k)
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ gtrnm1_ad(i,j,k)
    gtrnm1_ad(i,j,k) = 0.d0
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ab_gtr_ad(i,j) *abfac
    gtrnm1_ad(i,j,k) = gtrnm1_ad(i,j,k)-ab_gtr_ad(i,j) *abfac
    ab_gtr_ad(i,j) = 0.d0
  end do
end do
```

*File **ad_taf_output.f** has forward code & its adjoint all concatenated in one file*

Forward and Adjoint Code

How does TAF take the code on the left which is fairly straightforward to understand and transform it to the code on the right which is far less easy to interpret at first glance?

Forward model code snippet

```
do j = 1-oly, sny+oly
 do i = 1-olx, snx+olx
  ab_gtr(i,j) = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
  gtrnm1(i,j,k) = gtracer(i,j,k)
  gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
 end do
end do
```

Adjoint code equivalent generated by TAF

```
do j = 1-oly, sny+oly
 do i = 1-olx, snx+olx
  ab_gtr_ad(i,j) = ab_gtr_ad(i,j) +gtracer_ad(i,j,k)
  gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ gtrnm1_ad(i,j,k)
  gtrnm1_ad(i,j,k) = 0.d0
  gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ab_gtr_ad(i,j) *abfac
  gtrnm1_ad(i,j,k) = gtrnm1_ad(i,j,k)-ab_gtr_ad(i,j) *abfac
  ab_gtr_ad(i,j) = 0.d0
 end do
end do
```

*File **ad_taf_output.f** has forward code & its adjoint all concatenated in one file*

Consider each line of code as a subset of the full model ***M*** that advances a subset of the full model state vector *x as*

$x_i(t+1) = \boldsymbol{M}_i x_i(t)$

where *i* corresponds to a line of code, x_i is the subset of the full model state vector that appears in the source code line *i*, and $\boldsymbol{M}_i$ is the subset of the full model corresponding to source code line *i.*

Let us proceed by considering each line of code in turn, writing it in terms of

$x_i(t+1) = \boldsymbol{M}_i x_i(t)$

Note, each line of forward model code updates one variable but does not necessarily advance the model "calendar date" forward. Therefore, consider time levels *t* and *t* +1 as indicating the values of x_i before and after the operation of $\boldsymbol{M}_i$.

Forward model code snippet

```
ab_gtr(i,j)     = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
gtrnm1(i,j,k)   = gtracer(i,j,k)
gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
```

Step 1: Consider each line of code in turn, writing it in terms of $x_i(t+1) = \boldsymbol{M}_i\, x_i(t)$

ab_gtr = abfac*(gtracer -gtrnm1)

```
| gtracer   | = |1       0       0 | | gtracer |
| gtrnm1    |   |0       1       0 | | gtrmn1  |
| ab_gtr    |   |abfac -abfac    0 | | ab_gtr  |
```

x(t+1) = **M** x(t)

Forward model code snippet

```
ab_gtr(i,j)     = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
gtrnm1(i,j,k)   = gtracer(i,j,k)
gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
```

Step 1: Consider each line of code in turn, writing it in terms of $x_i(t+1) = \boldsymbol{M}_i\, x_i(t)$

gtrnm1(i,j,k) = gtracer(i,j,k)

```
| gtracer   | = |1     0 | | gtracer  |
| gtrnm1    |   |1     0 | | gtrmn1   |
```

x(t+1) = **M** x(t)

Forward model code snippet

```
ab_gtr(i,j)     = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
gtrnm1(i,j,k)   = gtracer(i,j,k)
gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
```

Step 1: Consider each line of code in turn, writing it in terms of $x_i(t+1) = \boldsymbol{M}_i\, x_i(t)$

gtracer(i,j,k) = gtracer(i,j,k) + ab_gtr(i,j)

```
| ab_gtr     |     |1     0 | | ab_gtr   |
| gtracer    | =   |1     1 | | gtracer  |
```

x(t+1) = **M** x(t)

Step 2: Line-by-line transformation of the forward model

Recall that
1. the adjoint of the linear model **M** is $\mathbf{M}^{\mathbf{T}}$
2. the adjoint model equivalent for a linear forward model of x(t) **=** $\mathbf{M}^{\mathbf{T}}$ x(t+1) is

x_ad(t) **=** $\mathbf{M}^{\mathbf{T}}$ x_ad(t+1)

- TAF uses the *"_ad"* suffix to indicate adjoint variables (sensitivities to J)

$$\frac{\partial J}{\partial x}(t) = \left(\left. \frac{\partial \mathbf{M}}{\partial x} \right|_t \right)^T \frac{\partial J}{\partial x}(t+1)$$

- note that time runs backwards in the adjoint model equation from *t* +1 to *t*

Forward model code snippet: Last line first

```
ab_gtr(i,j)     = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
gtrnm1(i,j,k)   = gtracer(i,j,k)
gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
```

```
gtracer(i,j,k) = gtracer(i,j,k) + ab_gtr(i,j)
```

forward model code snippet

```
| ab_gtr    |   |0    1 | | ab_gtr  |
| gtracer   | = |1    1 | | gtracer |
```

x(t+1) = **M** x(t)

forward model equation in matrix form

```
| ab_gtr_ad   |   | 0   1 | | ab_gtr_ad   |
| gtracer_ad  | = | 1   1 | | gtracer_ad  |
```

x_ad(t) = **M**^{T} x_ad(t+1)

adjoint model equation in matrix form

```
ab_gtr_ad   = ab_gtr_ad + gtracer_ad
gtracer_ad = gtracer_ad
```

adjoint model code snippet

```
gtracer(i,j,k) = gtracer(i,j,k) + ab_gtr(i,j)
```

forward model code snippet

```
ab_gtr_ad    = ab_gtr_ad + gtracer_ad
gtracer_ad   = gtracer_ad
```

adjoint model code snippet

Adjoint code equivalent generated by TAF

```
do j = 1-oly, sny+oly
  do i = 1-olx, snx+olx
    ab_gtr_ad(i,j) = ab_gtr_ad(i,j) +gtracer_ad(i,j,k)
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ gtrnm1_ad(i,j,k)
    gtrnm1_ad(i,j,k) = 0.d0
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ab_gtr_ad(i,j) *abfac
    gtrnm1_ad(i,j,k) = gtrnm1_ad(i,j,k)-ab_gtr_ad(i,j) *abfac
    ab_gtr_ad(i,j) = 0.d0
  end do
end do
```

TAF is smart enough to exclude the useless line "gtracer_ad = gtracer_ad" in the adjoint source code.

Forward model code snippet: analyze prior line

```
ab_gtr(i,j)     = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
gtrnm1(i,j,k)  = gtracer(i,j,k)
gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
```

```
gtrnm1(i,j,k) = gtracer(i,j,k)
```

forward model code snippet

```
 | gtracer   | = |1     0 | | gtracer |
 | gtrnm1    |   |1     0 | | gtrnm1  |
```

$$x(t+1) = \mathbf{M} \; x(t)$$

forward model equation in matrix form

```
| gtracer_ad | = | 1    1 | | gtracer_ad |
| gtrnm1_ad  |   | 0    0 | | gtrnm1_ad  |
```

$$x\_ad(t) = \mathbf{M}^T \; x\_ad(t+1)$$

adjoint model equation in matrix form

```
gtracer_ad = gtracer_ad + gtrnm1_ad
gtrnm1_ad    =   0
```

adjoint model code snippet

```
gtrnm1(i,j,k)  = gtracer(i,j,k)
```

forward model code snippet

```
gtracer_ad = gtracer_ad + gtrnm1_ad
gtrnm1_ad    =  0
```

adjoint model code snippet

Adjoint code equivalent generated by TAF

```
do j = 1-oly, sny+oly
  do i = 1-olx, snx+olx
    ab_gtr_ad(i,j) = ab_gtr_ad(i,j) +gtracer_ad(i,j,k)
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ gtrnm1_ad(i,j,k)
    gtrnm1_ad(i,j,k) = 0.d0
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ab_gtr_ad(i,j) *abfac
    gtrnm1_ad(i,j,k) = gtrnm1_ad(i,j,k)-ab_gtr_ad(i,j) *abfac
    ab_gtr_ad(i,j) = 0.d0
  end do
end do
```

Information about the sensitivity of J to past perturbations of `gtrnm1` *is lost because* `gtrnm1` *is replaced by* `gtracer` *in the forward code. This manifests in the adjoint code as* `gtracer_ad` *being set to zero.*

Forward model code snippet: analyze “first” line last

```
ab_gtr(i,j)      = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
gtrnm1(i,j,k)    = gtracer(i,j,k)
gtracer(i,j,k)   = gtracer(i,j,k)+ ab_gtr(i,j)
```

```
ab_gtr(i,j) = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
```

forward model code snippet

```
| gtracer   |     | 1         0       0 | | gtracer |
| gtrnm1    |  =  | 0         1       0 | | gtrmn1  |
| ab_gtr    |     | abfac  -abfac     0 | | ab_gtr  |
     x(t+1)          =          M                 x(t)
```

forward model equation in matrix form

```
| gtracer_ad  |     | 1    0    abfac | | gtracer_ad |
| gtrnm1_ad   |  =  | 0    1   -abfac | | gtrmn1_ad  |
| ab_gtr_ad   |     | 0    0    0     | | ab_gtr_ad  |
     x_ad(t)          =        MT            x_ad(t+1)
```

adjoint model equation in matrix form

```
gtracer_ad  = gtracer_ad +   ab_gtr_ad * abfac
gtrnm1_ad   = gtrmn1_ad  -   ab_gtr_ad * abfac
ab_gtr_ad   = 0
```

adjoint model code snippet

```
ab_gtr(i,j) = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
```

forward model code snippet

```
gtracer_ad  = gtracer_ad +   ab_gtr_ad * abfac
gtrnm1_ad   = gtrmn1_ad  -   ab_gtr_ad * abfac
ab_gtr_ad   = 0
```

adjoint model code snippet

Adjoint code equivalent generated by TAF

```
do j = 1-oly, sny+oly
 do i = 1-olx, snx+olx
  ab_gtr_ad(i,j) = ab_gtr_ad(i,j) +gtracer_ad(i,j,k)
  gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ gtrnm1_ad(i,j,k)
  gtrnm1_ad(i,j,k) = 0.d0
  gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ab_gtr_ad(i,j) *abfac
  gtrnm1_ad(i,j,k) = gtrnm1_ad(i,j,k)-ab_gtr_ad(i,j) *abfac
  ab_gtr_ad(i,j) = 0.d0
 end do
end do
```

Summary: Line-by-line transformation of the forward model is interpretable

1. Generation of adjoint code occurs via line-by-line transformation on the forward code
2. Consider each **line of model code** as a ‘mini model’ that operates on an small subset of the full state vector

$$x(t) = \mathbf{M}^{\mathbf{T}} x(t+1)$$

3. Adjoint model source code is created ***in reverse order***
   - the adjoint of a linear model **M** is $\mathbf{M}^{\mathbf{T}}$
   - the adjoint model equation is

$$x\_ad(t) = \mathbf{M}^{\mathbf{T}} x\_ad(t+1)$$

Forward and Adjoint Code

Forward model code snippet

```
do j = 1-oly, sny+oly
  do i = 1-olx, snx+olx
    ab_gtr(i,j) = abfac*(gtracer(i,j,k)-gtrnm1(i,j,k))
    gtrnm1(i,j,k) = gtracer(i,j,k)
    gtracer(i,j,k) = gtracer(i,j,k)+ ab_gtr(i,j)
  end do
end do
```

Adjoint code equivalent generated by TAF

```
do j = 1-oly, sny+oly
  do i = 1-olx, snx+olx
    ab_gtr_ad(i,j) = ab_gtr_ad(i,j) +gtracer_ad(i,j,k)
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ gtrnm1_ad(i,j,k)
    gtrnm1_ad(i,j,k) = 0.d0
    gtracer_ad(i,j,k) = gtracer_ad(i,j,k)+ab_gtr_ad(i,j) *abfac
    gtrnm1_ad(i,j,k) = gtrnm1_ad(i,j,k)-ab_gtr_ad(i,j) *abfac
    ab_gtr_ad(i,j) = 0.d0
  end do
end do
```

*File **ad_taf_output.f** has forward code & its adjoint all concatenated in one file*

What happens when the model M is nonlinear?

$$x(t+1) = \mathbf{M}(\mathbf{x}(\mathbf{t}))$$

$$x_{ad}(t) = \left(\frac{\partial \mathrm{M}}{\partial x} \bigg|_t \right)^T x_{ad}(t+1)$$

z = a z^3 + b z + y

hypothetical forward model code snippet

$$\begin{vmatrix} y \\ z \end{vmatrix} = \begin{vmatrix} y & 0 \\ y & az^3+bz \end{vmatrix}$$

x(t+1) = **M(x(t))**

forward model equation in matrix form

What is the mini adjoint model in matrix form?

What is the adjoint model code snippet corresponding with

What happens when the model M is nonlinear?

$$x(t+1) = \mathbf{M}(\mathbf{x}(\mathbf{t})))$$

$$x_{ad}(t) = \left(\frac{\partial \mathrm{M}}{\partial x}\bigg|_t \right)^T x_{ad}(t+1)$$

z = a z^3 + b z + y

hypothetical forward model code snippet

$$\begin{vmatrix} y \\ z \end{vmatrix} = \begin{vmatrix} y & 0 \\ y & az^3+bz \end{vmatrix}$$

x(t+1) = **M(x(t))**

forward model equation in matrix form

$$\begin{vmatrix} y\_ad \\ z\_ad \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 0 & 3az^2+b \end{vmatrix} \begin{vmatrix} y\_ad \\ z\_ad \end{vmatrix}$$

x_ad(t) **=** $\left(\frac{\partial \mathrm{M}}{\partial x}\bigg|_t \right)^T$ x_ad(t+1)

adjoint model equation in matrix form