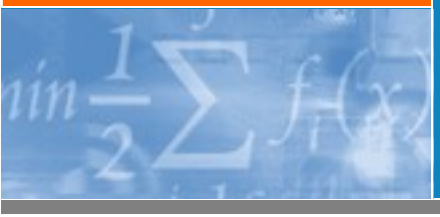
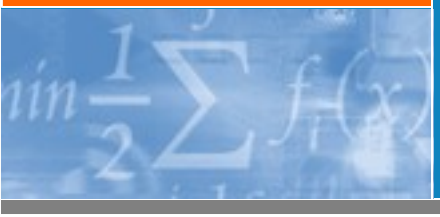


The Future of Fortran is Bright …

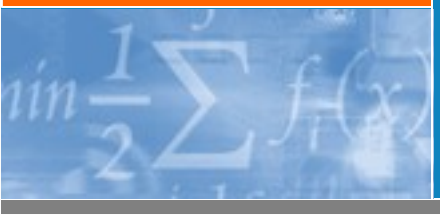
Fortran @ 50

Richard Hanson



Samples -

- Sparse linear systems
- Differential-algebraic equations
- Parallel Programming with Co-array Fortran



- ***Sparse linear systems:***

  real :: y(:), b(:)

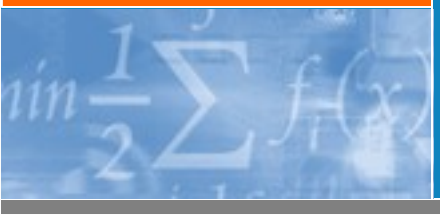
  Type(Sparse) A

  …

  y = A .ix. b

Uses derived types and a defined operation, .ix. An LU factorization plus iterative refinement computes the y in a hidden routine.

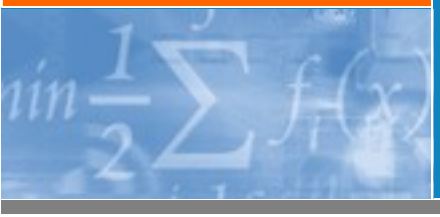
$$i.e. \quad y = A^{-1}b$$



- ***Differential – algebraic systems:***

Solve a system of equations

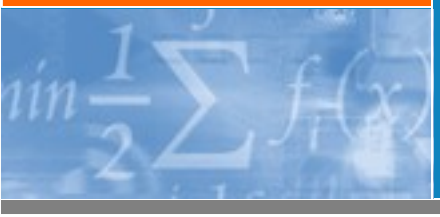
$$F(y, y', t) = 0,\ y(t_0) = y_0,\ y' = \frac{dy}{dt}$$



Solver needs user input of the math objects in the form of subprograms for:

$$F(y, y', t), y_0, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial y'}$$

Typically need data with these math objects and may also require optional linear equation solvers and numerical differentiation routines based on the problem size and structure

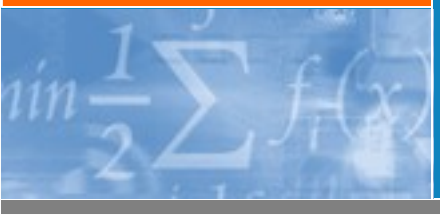


The DAE solver uses a packaged derived type:

TYPE(DAE_Dope) DAE_Type

The user ***extends*** this type to include problem data:

```
TYPE, extends (DAE_Dope) My_DAE_Type
   real :: Problem_Data(100)
End Type My_DAE_Type
```

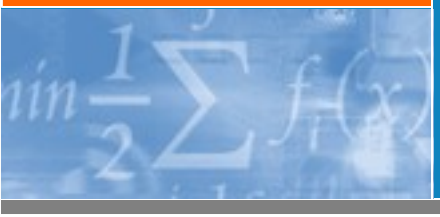


The DAE solver and the user-written routine for the math objects refer to the extended derived type as a *polymorphic argument*: Here is an outline of the user routine:

```
Subroutine F(Y,Yprime,t, DAE_Stuff)
    ...
    Class(DAE_Dope), intent(InOut) :: DAE_Stuff
```

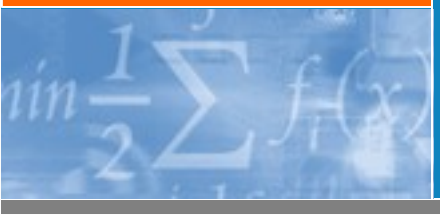
(Continued …)

```
! Unravel the class object to get user data –
   Select Type(DAE_Stuff)

   Type is (My_DAE_Type)
    … = DAE_Stuff % Program_Data(:)

End Select

! Compute F(y,y',t) …
    F = …

End Subroutine F
```



- ***Parallel Programming with Co-array Fortran***

Syntax is introduced for defining an executing *image* of a program:




```
Type Phu
   Real R
   Character(len=10) scribble
End Type Phu
```

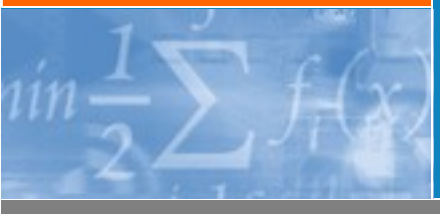
Type(Phu) :: z[*] ! Note the brackets[ ] - z is a co-array

Sync all

```
if (this_image() == 1) then
      read(*,*) z
      do image = 2, num_images()
          z[image] = z ! Broadcast z to all other images
      end do
end if
```

Sync all

! The integer intrinsics *this_image() and num_images()*
! are in the F2008 standard proposal. *Sync all* is a
! key word that synchronizes the executing program images.



Other notable features of Modern Fortran –

- Interoperability with C
- IEEE floating point exception handling
- Procedure pointers
- Associate constructs