A MATLAB to Fortran conversion "Manual" [1]

General remarks and programming

	MATLAB	FORTRAN90
Editor	Built-in	Built-in
Compilation	Run-time compilation	Compile by `Ctrl+Shift+b` and run by `Ctrl+F5`
Cases	Case sensitive ($K \neq k$)	Case in-sensitive ($K = k$)
Variables	Scalar variables need not be defined	All variables must be defined: integer :: n real(wp) :: length real(wp) :: d, f(17), kmatrix(13,13) real(wp), dimension(17) :: f character(len = 6) :: word logical :: banded
Dynamic allocation	NA	real(wp), dimension(:,:), allocatable :: kmatrix ⋮ allocate (kmatrix(neqn, neqn))
Modules	NA	May contain "private" and "public" subroutines
Types	We did not use it	see under "FEM specific remarks"
Subroutine calls	[areas] = bisect(areas,amin, ...); function [areas] = bisect(aold,amin, ...) ⋮	call bisect(areas, amin, ...) subroutine bisect(aold, amin, ...) real(wp), intent(in) :: amin, ... real(wp), dimension(:), intent(inout) :: aold ⋮

Programming

	MATLAB	FORTRAN90
do loops	*for* $i = 1 : n$ ... *end*	*do* $i = 1, n$ ... *end do*

if statements	*if* (...)
	...
	end
		if (...) *then*
		...
		end if

Indexing	K(edof,edof) = K(edof,edof) + k	kmatrix(edof,edof) = kmatrix(edof,edof) + ke
Vector multiplication	a = b'*c	a = dot_product(b,c)
Matrix multiplication	a = b*c	a = matmul(b,c)
printing to screen	remove ";"	print*,'aout=',aout
print matrix	K	do i = 1,neqn print "(24(f4.2,tr1))",kmatrix(i,1:neqn) end do
# of colums in matrix	size(K,2)	size(kmatrix,2)
Assigning variables	i = 1500 d = 1500 d = 1500 d = 1500 banded = 0 (or 1) word = 'blabla'	i = 1500 [integer :: i] d = 1500.0 (or 1.5e3) [real :: d] d = 1500.d0 (or 1.5d3) [real(wp) :: d] - legacy notation d = 1500.0_wp (or 1.5e3_wp) [real(wp) :: d] - modern not. banded = .false. [or .true.] (logical) word = 'blabla' (character)
Integer division	i=1, j=3 => i/j = 0.3333	i=1, j=3 => i/j = 0 (NB! Integer part of result!)

[1] Written for the course FEM-Heavy, Fall 2018 by Ole Sigmund.

FEM-Heavy-specific remarks

	MATLAB	FORTRAN90
Number of equations	neqn	neqn
Number of elements	ne	ne
Topology matrix	IX	ix
Nodal coordinate matrix	X	x
Element vector (type)	NA NA NA IX(e,3)	element(e)%numnode number of nodes in element e element(e)%id element type of element e $= \begin{cases} 1 & truss \\ 2 & 4-node \end{cases}$ element(e)%ix(i) node i of element e element(e)%mat material property number of element e
Number of nodes pr. element	2	nen=element(e)%numnode
X-coord. of node 2 in element e	X(IX(e,2),1)	x(element(e)%ix(2),1)
Material property matrix (type)	Young's mod.: mprop(IX(e,3),1) Area: mprop(IX(e,3),2)	mprop(element(e)%mat)%young mprop(element(e)%mat)%area mprop(element(e)%mat)%nu mprop(element(e)%mat)%thk mprop(element(e)%mat)%dens mprop(element(e)%mat)%youngy mprop(element(e)%mat)%shear
number of boundary conditions	size(bound,1)	nb
boundary cond. matrix	bound(:,3)	bound(1:nb,1:3)
number of loads	size(loads,1)	np
load matrix	loads(:,1:3) (node,dof,force)	loads(1:np,1:4) load-type=1: point load: (load-type,node,dof,force) load-type=2: pressure load: (load-type,element,face,pressure)
equation solving	D = K \ P	call factor(kmatrix) call solve(kmatrix,p) d(1:neqn)=p(1:neqn)