

```

program lid_driven_cavity
implicit none
integer, parameter :: N = 50 ! grid size (N x N grid)
real :: dx, dy, dt, Re ! grid spacing, time step, Reynolds number
real :: u(N), v(N), p(N), ! velocity and pressure fields
integer :: step
real :: start_time, end_time, elapsed_time

```

! Parameters

```

dx = 1.0 / (N-1) ! Grid spacing in x direction
dy = 1.0 / (N-1) ! Grid spacing in y direction
dt = 0.001 ! time step size
Re = 100 ! Reynolds number

```

! Initialize arrays

```

u = 0.0
v = 0.0
p = 0.0

```

! Initialize the top boundary (lid) velocity

```

u(N,:) = 1.0

```

! Start timing

```

call cpu_time(start_time)

```

! Main loop for time stepping

```

do step = 1, 1000
  call compute_velocity(u, v, p, dx, dy, dt, Re)
  call update_pressure(p, dx, dy)

```

```

  ! Output or check convergence
  if (mod(step, 100) == 0) then
    print *, Step: , step
  end if
end do

```

! Stop timing

```

call cpu_time(end_time)
elapsed_time = end_time - start_time
print *, Elapsed time for CAD simulation: , elapsed_time, ' seconds'

```

contains

! Function to update the velocity and pressure fields (simplified)

```

subroutine compute_velocity(u, v, p, dx, dy, dt, Re)

```

```

real, dimension(:,:), intent(inout) :: u, v, p
real, intent(in) :: dx, dy, dt, Re
integer :: i, j

```

! Simple explicit method for velocity (simplified)

```

do i = 2, N-1
  do j = 2, N-1
    u(i, j) = u(i, j) - dt * ((u(i, j) * (u(i+1, j) - u(i-1, j))) / (2 * dx) - u(i, j-1) * (u(i, j+1) - u(i, j-1))) / (2 * dy)
  end do
end do

```

end do

! Simple velocity update for v (similar)

```

do i = 2, N-1
  do j = 2, N-1

```

```

1      v(i, j) = v(i, j) - dt * ((u(i, j) * (v(i+1, j) - v(i-1, j))) / (2 * dx)
2                                     (v(i, j+1) - v(i, j-1))) / (2 * dy)
3
4      end do
5  end subroutine compute_velocity
6
7  ! Function to solve for pressure (simplified Poisson equation solver)
8  subroutine update_pressure(p, dx, dy)
9      real, dimension(:,:), intent(inout) :: p
10     real, intent(in) :: dx, dy
11     integer :: i, j
12
13     ! Simple pressure Poisson equation (Jacobi iteration)
14     do i = 1, N-1
15         do j = 1, N-1
16             p(i, j) = 0.25 * (p(i+1, j) + p(i-1, j) + p(i, j+1) + p(i, j-1))
17         end do
18     end do
19 end subroutine update_pressure
20
21 end program lid_driven_cavity

```