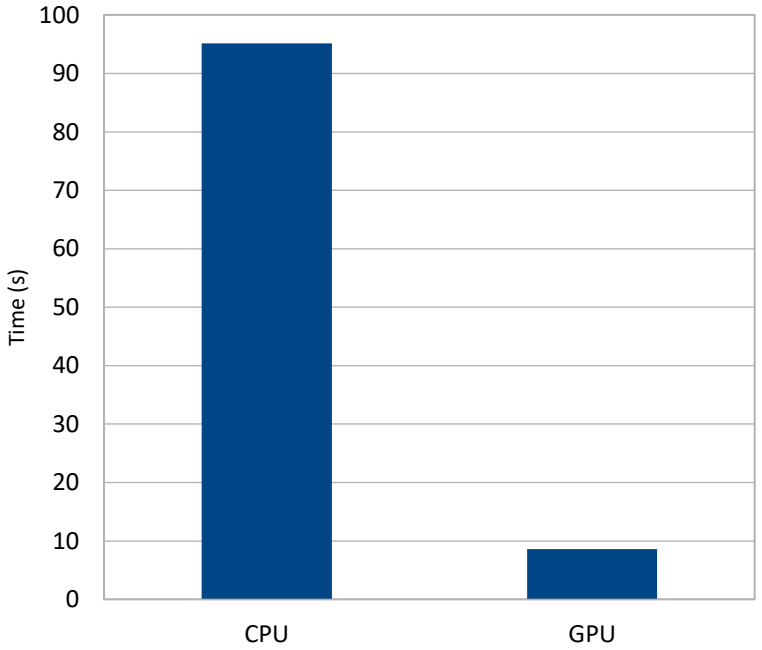
Speed up of matrix assembling on GPU



Comparison of code before and after optimization

```
function v1ubb(e,f,g,h)
  use basic
  use m3dc1_nint

  implicit none

  vectype, dimension(dofs_per_element) :: v1ubb
  vectype, intent(in), dimension(dofs_per_element,MAX_PTS,OP_NUM) :: e
  vectype, intent(in), dimension(MAX_PTS,OP_NUM) :: f,g,h

  vectype, dimension(dofs_per_element) :: temp
  vectype, dimension(dofs_per_element,MAX_PTS) :: tempa
  integer :: j

  temp = 0.

     if(surface_int) then
        temp79a = f(:,OP_DZ)*g(:,OP_DR) - f(:,OP_DR)*g(:,OP_DZ)
        temp = intx4(e(:,:,OP_1),temp79a,norm79(:,2),h(:,OP_DR)) &
             - intx4(e(:,:,OP_1),temp79a,norm79(:,1),h(:,OP_DZ))
#if defined(USE3D) || defined(USECOMPLEX)
        temp79a = ri2_79*h(:,OP_DP)
        temp = temp &
             + intx5(e(:,:,OP_1),temp79a,norm79(:,1),f(:,OP_DR),g(:,OP_DP)) &
             + intx5(e(:,:,OP_1),temp79a,norm79(:,2),f(:,OP_DZ),g(:,OP_DP)) &
             + intx5(e(:,:,OP_1),temp79a,norm79(:,1),f(:,OP_DRP),g(:,OP_1)) &
             + intx5(e(:,:,OP_1),temp79a,norm79(:,2),f(:,OP_DZP),g(:,OP_1))
#endif
     else
#if defined(USE3D) || defined(USECOMPLEX)
        do j=1, dofs_per_element
           tempa(j,:) = &
                (e(j,:,OP_DZ)*f(:,OP_DZPP) + e(j,:,OP_DR)*f(:,OP_DRPP)) &
                *g(:,OP_1) &
                + 2.*(e(j,:,OP_DZ)*f(:,OP_DZP) + e(j,:,OP_DR)*f(:,OP_DRP)) &
                *g(:,OP_DP) &
                +    (e(j,:,OP_DZ)*f(:,OP_DZ) + e(j,:,OP_DR)*f(:,OP_DR)) &
                *g(:,OP_DPP)
        end do
        temp = intx3(tempa,ri2_79,h(:,OP_1))
#endif
     end if

  v1ubb = temp
end function v1ubb
```

```
function v1ubb(g,h)
  use basic
  use m3dc1_nint

  implicit none

  type(prodarray) :: v1ubb
  vectype, intent(in), dimension(MAX_PTS,OP_NUM) :: g,h

  type(prodarray) :: temp, tempb
  type(muarray) :: tempa
  integer :: j

  temp%len = 0.

     if(surface_int) then
        tempa = mu(g(:,OP_DR),OP_DZ) + mu(-g(:,OP_DZ),OP_DR)
        temp = prod(mu( norm79(:,2)*h(:,OP_DR),OP_1),tempa) &
             + prod(mu(-norm79(:,1)*h(:,OP_DZ),OP_1),tempa)
#if defined(USE3D) || defined(USECOMPLEX)
        temp79a = ri2_79*h(:,OP_DP)
        temp = temp &
             + prod(temp79a*norm79(:,1)*g(:,OP_DP),OP_1,OP_DR) &
             + prod(temp79a*norm79(:,2)*g(:,OP_DP),OP_1,OP_DZ) &
             + prod(temp79a*norm79(:,1)*g(:,OP_1),OP_1,OP_DRP) &
             + prod(temp79a*norm79(:,2)*g(:,OP_1),OP_1,OP_DZP)
#endif
     else
#if defined(USE3D) || defined(USECOMPLEX)
        tempb = prod(g(:,OP_1),OP_DZ,OP_DZPP) + prod(g(:,OP_1),OP_DR,OP_DRPP) &
              + prod(2.*g(:,OP_DP),OP_DZ,OP_DZP) + prod(2.*g(:,OP_DP),OP_DR,OP_DRP) &
              + prod(g(:,OP_DPP),OP_DZ,OP_DZ) + prod(g(:,OP_DPP),OP_DR,OP_DR)
        temp = tempb*(ri2_79*h(:,OP_1))
#endif
     end if

  v1ubb = temp
end function v1ubb
```

Definition of derived data type and operators

```
type muarray
     sequence
     integer :: len
     vectype, dimension(MAX_PTS, OP_NUM) :: value79
     integer, dimension(OP_NUM) :: op
  end type muarray

  type prodarray
     sequence
     integer :: len
     vectype, dimension(MAX_PTS, OP_NUM*OP_NUM) :: value79
     integer, dimension(OP_NUM*OP_NUM) :: op1
     integer, dimension(OP_NUM*OP_NUM) :: op2
  end type prodarray

  interface operator(+)
     module procedure prod_add
     module procedure mu_add
  end interface

  interface operator(*)
     module procedure mu_mult_array
     module procedure prod_mult_array
  end interface

  interface mu
     module procedure mu_new_array
  end interface

  interface prod
     module procedure prod_mu_mu
     module procedure prod_new_array
  end interface
```