

```
[> restart:
```

Code updated on 04/23/2022 to report cpu time taken and memory used using CodeTools:-Usage. **Code is released under MIT license**

04/21/2022. Code written by Dr. Venkat Subramanian and MAPLE group at UT. The code is provided as open-access with no restrictions.

This code is very similar to the standard 2D Battphase code. The only difference is the Pardiso linear solver is called after changing the dll files. Both the clock time and total computation time are reported. CPU reported in Table 3 at the end of the code for 160 node points in each direction. The entire Jacobian is not used. Only the non-zero sparse entries are identified (only once) and updated with time.

```
> t11:=time[real]():t12:=time():
```

```
> gc();
```

```
> Digits:=15;
```

```
Digits := 15
```

(1)

```
> interface(prettyprint=2);
```

```
2
```

(2)

```
> EFAdd:=proc(Y00::Matrix(datatype=float[8]),Ymid::Matrix(datatype=
float[8]),Phi0::Vector(datatype=float[8]),F0::Matrix(datatype=
float[8]),dt::float[8],N::integer,M::integer,e1::float[8])
```

```
local i::integer,j::integer;
```

```
for i from 2 to N-1 do for j from 2 to M-1 do Ymid[i,j]:=max(e1,
Y00[i,j]-dt*F0[i,j]*Phi0[i+(j-1)*N]):od:od:
```

```
for i from 1 to N do Ymid[i,1]:=Ymid[i,2]:Ymid[i,M]:=Ymid[i,M-1]:
od:
```

```
for j from 1 to M do Ymid[1,j]:=Ymid[2,j]:Ymid[N,j]:=Ymid[N-1,j]:
od:
```

```
end proc:
```

```
> EFAdd:=Compiler:-Compile(EFAdd):
```

```
> EFAdd2:=proc(Y00::Matrix(datatype=float[8]),Ymid::Matrix
(datatype=float[8]),Phi0::Vector(datatype=float[8]),F0::Matrix
(datatype=float[8]),dt::float[8],N::integer,M::integer,e1::float
[8])
```

```
local i::integer,j::integer;
```

```
for i from 2 to N-1 do for j from 2 to M-1 do
```

```
Ymid[i,j]:=max(e1,Y00[i,j]*3/4.+Ymid[i,j]/4.-dt/4.*F0[i,j]*Phi0
[i+(j-1)*N]):od:od:
```

```
for i from 1 to N do Ymid[i,1]:=Ymid[i,2]:Ymid[i,M]:=Ymid[i,M-1]
:od:
```

```
for j from 1 to M do Ymid[1,j]:=Ymid[2,j]:Ymid[N,j]:=Ymid[N-1,j]
```

```
:od:  
end proc:
```

```
> EFAdd2:=Compiler:-Compile(EFAdd2):
```

```
> EFAdd3:=proc(Y00::Matrix(datatype=float[8]),Ymid::Matrix  
(datatype=float[8]),Phi0::Vector(datatype=float[8]),F0::Matrix  
(datatype=float[8]),dt::float[8],N::integer,M::integer,e1::float  
[8])  
local i::integer,j::integer;  
for i from 2 to N-1 do for j from 2 to M-1 do  
Y00[i,j]:=max(e1,Y00[i,j]*1/3.+Ymid[i,j]*2/3.-dt*2/3.*F0[i,j]*  
Phi0[i+(j-1)*N]):od:od:  
for i from 1 to N do Y00[i,1]:=Y00[i,2]:Y00[i,M]:=Y00[i,M-1]:od:  
for j from 1 to M do Y00[1,j]:=Y00[2,j]: Y00[N,j]:=Y00[N-1,j]:od:  
end proc:
```

```
> EFAdd3:=Compiler:-Compile(EFAdd3):
```

```
> YdatStore:=proc(Y00::Matrix(datatype=float[8]),Ydat::Array  
(datatype=float[8]),N::integer,M::integer,jj::integer)  
local i::integer,j::integer;  
for i from 2 to N-1 do for j from 2 to M-1 do  
Ydat[jj,i,j]:=Y00[i,j]:od:od:  
end proc:
```

```
> YdatStore:=Compiler:-Compile(YdatStore):
```

```
> phiaveAdd:=proc(Y00::Matrix(datatype=float[8]),N::integer,  
M::integer,Ny::integer)  
local i::integer,j::integer,phiave::float;  
phiave:=0.0:  
for i from 2 to N-1 do for j from 2 to M-1 do phiave:=phiave+Y00  
[i,j]:od:od:  
phiave/(N-2)/(M-2);  
end proc:
```

```
> phiaveAdd:=Compiler:-Compile(phiaveAdd):
```

```
> PhiAdd:=proc(N::integer,Phi0::Vector(datatype=float[8]),  
db::Vector(datatype=float[8]))  
local i::integer;  
for i from 1 to N do Phi0[i]:=Phi0[i]+db[i]:od:  
end proc:
```

```
> PhiAdd:=Compiler:-Compile(PhiAdd):
```

```

> PhiSwitch:=proc(N::integer,Phi0::Vector(datatype=float[8]),
  Phitemp::Vector(datatype=float[8]))
  local i::integer;
  for i from 1 to N do Phitemp[i]:=Phi0[i]:od:
end proc:
=
> PhiSwitch:=Compiler:-Compile(PhiSwitch):
=
> MidPred:=proc(N::integer,Phi0::Vector(datatype=float[8]),
  Phitemp::Vector(datatype=float[8]),Phimid::Vector(datatype=float
  [8]))
  local i::integer;
  for i from 1 to N do Phimid[i]:=Phitemp[i]/2.0+Phi0[i]/2.0:od:
end proc:
=
> MidPred:=Compiler:-Compile(MidPred):
=
> y0proc:=proc(NN::integer,Y00::Matrix(datatype=float[8]),
  beta::float[8],e1::float[8])
  local i,j,xx,yy,N,h,w,MM,M,Ny,rf,f,ff,rr;
> N:=NN+2:h:=1.0/NN:w:=h*beta:
> MM:=NN;M:=MM+2;Ny:=MM;
> for i from 2 to N-1 do for j from 2 to M-1 do
  xx:=-0+(i-1/2-1)*h:yy:=-0+(j-1/2-1)*h:
  rr:=xx^2+yy^2;
  #if j<4 and i <N/5+1 then Y00[i,j]:=1e-9 else Y00[i,j]:=1.0:end:
  #Y00[i,j]:=1.0:
  #if rr <9/100. then Y00[i,j]:=1e-12; else Y00[i,j]:=1.0:end:
  #if rr=9/100. then Y00[i,j]:=0.5:end:
  Y00[i,j]:=max(e1,0.5+0.5*tanh((sqrt(rr)-0.3)/w/sqrt(2.0))):
  #if abs(xx-0.5)>=0.25 and yy<=0.5 then Y00[i,j]:=1e-9 else Y00[i,
  j]:=1.0:end:
  #if yy<=0.1 then Y00[i,j]:=1e-9 :end:
  od:od:
  for i from 1 to N do
  Y00[i,1]:=Y00[i,2]:
  Y00[i,M]:=Y00[i,M-1]:
  od:
  for j from 1 to M do
  Y00[1,j]:=Y00[2,j]:
  Y00[N,j]:=Y00[N-1,j]:
  od:
end proc:
=
> y0proc0:=proc(NN,Y00)
  local i,j,xx,yy,N,h,w,MM,M,Ny,rf,f,ff,rr;

```

```

> N:=NN+2:h:=1.0/NN:w:=h/2.0:
> MM:=NN;M:=MM+2;Ny:=MM;
> for i from 2 to N-1 do for j from 2 to M-1 do
  xx:=-0+(i-1/2-1)*h:yy:=-0+(j-1/2-1)*h:
  rr:=xx^2+yy^2;
  #if j<4 and i <N/5+1 then Y00[i,j]:=1e-9 else Y00[i,j]:=1.0;end:
  #Y00[i,j]:=1.0:
  #if rr <9/100. then Y00[i,j]:=1e-12; else Y00[i,j]:=1.0:end:
  #if rr=9/100. then Y00[i,j]:=0.5:end:
  #Y00[i,j]:=max(1e-12,0.5+0.5*tanh((sqrt(rr)-0.3)/w/sqrt(2.0))):
  if abs(xx-0.5)>=0.25 and yy<=0.5 then Y00[i,j]:=1e-9 else Y00[i,
  j]:=1.0:end:
  if yy<=0.1 then Y00[i,j]:=1e-9 :end:
  od:od:
  for i from 1 to N do
  Y00[i,1]:=Y00[i,2]:
  Y00[i,M]:=Y00[i,M-1]:
  od:
  for j from 1 to M do
  Y00[1,j]:=Y00[2,j]:
  Y00[N,j]:=Y00[N-1,j]:
  od:
  end proc:
> y0proc:=Compiler:-Compile(y0proc):
> printlevel:=2;

```

printlevel := 2

(3)

```

> Eqs11:=proc(N::integer,M::integer,Y0::Matrix(datatype=float[8]),
  F::Matrix(datatype=float[8]),y::Vector(datatype=float[8]),
  v0::float,ff::Vector(datatype=float[8]))
  local i::integer,j::integer,i1::integer,h::float[8];
  h:=1.0/(N-2):
  j:=1: for i from 1 to N do
  i1:=i+(j-1)*N:
  ff[i1]:=-y[i1]+y[i1+N]:
  od:

  for j from 2 to M-1 do
  i:=1:
  i1:=i+(j-1)*N:
  ff[i1]:=-y[i1]+y[i1+1]:
  for i from 2 to N-1 do

```

```

i1:=i+(j-1)*N:
ff[i1]:=
(Y0[i,j]+Y0[i,j+1])*(y[i1+N]-y[i1])
-(Y0[i,j]+Y0[i,j-1])*(y[i1]-y[i1-N])
+(Y0[i+1,j]+Y0[i,j])*(y[i1+1]-y[i1])
-(Y0[i,j]+Y0[i-1,j])*(y[i1]-y[i1-1])
-y[i1]*h*(1e-24+(Y0[i+1,j]-Y0[i-1,j])^2+(Y0[i,j+1]-Y0[i,j-1])^2)^
(1/2):
od:
i:=N:
i1:=i+(j-1)*N:
ff[i1]:=-y[i1]+y[i1-1]+0*h:
od:

j:=M:
for i from 1 to N do
i1:=i+(j-1)*N:
ff[i1]:=-y[i1]+y[i1-N]+v0*h:
od:

end proc:

```

```
> Eqs11:=Compiler:-Compile(Eqs11):
```

```

> XX:=proc(N::integer,M::integer,Y0::Matrix(datatype=float[8]),
y::Vector(datatype=float[8]),v0::float,x0::Vector(datatype=float
[8]))
local i::integer,j::integer,i1::integer,h::float[8],
count::integer;
h:=1.0/(N-2):
count:=0:
j:=1:
for i from 1 to N do
i1:=i+(j-1)*N:
#ff[i1]:=-y[i1]+y[i1+N]:
count:=count+1:
x0[count]:=1.0:
count:=count+1:
x0[count]:=-1.0:
#j00[i1,i1]:=1.0:j00[i1,i1+N]:=-1.00:
od:

for j from 2 to M-1 do

```

```

i:=1:
i1:=i+(j-1)*N:
count:=count+1:
x0[count]:=1.0:
count:=count+1:
x0[count]:=-1.0:
#j00[i1,i1]:=1.0:j00[i1,i1+1]:=-1.00:
for i from 2 to N-1 do
i1:=i+(j-1)*N:
count:=count+1:
x0[count]:=-Y0[i,j-1]-Y0[i,j]:
count:=count+1:
x0[count]:=-Y0[i-1,j]-Y0[i,j]:
count:=count+1:
x0[count]:=4*Y0[i,j]+Y0[i,j+1]+Y0[i,j-1]+Y0[i+1,j]+Y0[i-1,j]+h*
(1e-24+(Y0[i+1,j]-Y0[i-1,j])^2+(Y0[i,j+1]-Y0[i,j-1])^2)^(1/2):
count:=count+1:
x0[count]:=-Y0[i+1,j]-Y0[i,j]:
count:=count+1:
x0[count]:=-Y0[i,j+1]-Y0[i,j]:
#j00[i1,i1]:=4*Y0[i,j]+Y0[i,j+1]+Y0[i,j-1]+Y0[i+1,j]+Y0[i-1,j]+h*
(1e-24+(Y0[i+1,j]-Y0[i-1,j])^2+(Y0[i,j+1]-Y0[i,j-1])^2)^(1/2):
#j00[i1,i1+1]:=-Y0[i+1,j]-Y0[i,j]:j00[i1,i1-1]:=-Y0[i-1,j]-Y0[i,
j]:
#j00[i1,i1+N]:=-Y0[i,j+1]-Y0[i,j]:j00[i1,i1-N]:=-Y0[i,j-1]-Y0[i,
j]:
od:
i:=N:
count:=count+1:
x0[count]:=-1.0:
count:=count+1:
x0[count]:=1.0:
i1:=i+(j-1)*N:
#j00[i1,i1]:=1.0:j00[i1,i1-1]:=-1.00:
od:

j:=M:
for i from 1 to N do
i1:=i+(j-1)*N:
count:=count+1:
x0[count]:=-1.0:
count:=count+1:

```

```
x0[count]:=1.0:
#j00[i1,i1]:=1.0:j00[i1,i1-N]:=-1.00:
#ff[i1]:=-y[i1]+y[i1-N]+v0*h:
od:
```

```
end proc:
```

```
> XX:=Compiler:-Compile (XX) :
```

```
> CBproc:=proc (N::integer,M::integer,CB0::Vector (datatype=integer
[4]))
```

```
local i::integer,j::integer;
```

```
CB0[1]:=1:
```

```
> for i from 1 to N do CB0[i+1]:=CB0[i]+2:od:
```

```
> for j from 2 to M-1 do
```

```
  i:=1:CB0[i+(j-1)*N+1]:=CB0[i+(j-1)*N]+2:
```

```
  for i from 2 to N-1 do
```

```
    CB0[i+(j-1)*N+1]:=CB0[i+(j-1)*N]+5:od:
```

```
  i:=N:CB0[i+(j-1)*N+1]:=CB0[i+(j-1)*N]+2:od:
```

```
> for i from 1 to N do CB0[N*M-N+1+i]:=CB0[N*M-N+i]+2:od:
```

```
end proc:
```

```
> CBproc:=Compiler:-Compile (CBproc) :
```

```
> Rproc:=proc (N::integer,M::integer,R0::Vector (datatype=integer [4]
)
)
```

```
local i::integer,j::integer,i1::integer,count::integer;
```

```
count:=0:
```

```
j:=1:
```

```
for i from 1 to N do
```

```
  i1:=i+(j-1)*N:
```

```
  #ff[i1]:=-y[i1]+y[i1+N]:
```

```
  count:=count+1:
```

```
  R0[count]:=i1:
```

```
  count:=count+1:
```

```
  R0[count]:=i1+N:
```

```
  #j00[i1,i1]:=1.0:j00[i1,i1+N]:=-1.00:
```

```
od:
```

```
for j from 2 to M-1 do
```

```
  i:=1:
```

```
  i1:=i+(j-1)*N:
```

```
  count:=count+1:
```

```
  R0[count]:=i1:
```

```
  count:=count+1:
```

```

R0[count]:=i1+1:
#j00[i1,i1]:=1.0:j00[i1,i1+1]:=-1.00:
for i from 2 to N-1 do
i1:=i+(j-1)*N:
count:=count+1:
R0[count]:=i1-N:
count:=count+1:
R0[count]:=i1-1:
count:=count+1:
R0[count]:=i1:
count:=count+1:
R0[count]:=i1+1:
count:=count+1:
R0[count]:=i1+N:
#j00[i1,i1]:=4*Y0[i,j]+Y0[i,j+1]+Y0[i,j-1]+Y0[i+1,j]+Y0[i-1,j]+h*
(1e-24+(Y0[i+1,j]-Y0[i-1,j])^2+(Y0[i,j+1]-Y0[i,j-1])^2)^(1/2):
#j00[i1,i1+1]:=-Y0[i+1,j]-Y0[i,j]:j00[i1,i1-1]:=-Y0[i-1,j]-Y0[i,
j]:
#j00[i1,i1+N]:=-Y0[i,j+1]-Y0[i,j]:j00[i1,i1-N]:=-Y0[i,j-1]-Y0[i,
j]:
od:
i:=N:
i1:=i+(j-1)*N:
count:=count+1:
R0[count]:=i1-1:
count:=count+1:
R0[count]:=i1:
i1:=i+(j-1)*N:
#j00[i1,i1]:=1.0:j00[i1,i1-1]:=-1.00:
od:

j:=M:
for i from 1 to N do
i1:=i+(j-1)*N:
count:=count+1:
R0[count]:=i1-N:
count:=count+1:
R0[count]:=i1:
#j00[i1,i1]:=1.0:j00[i1,i1-N]:=-1.00:
#ff[i1]:=-y[i1]+y[i1-N]+v0*h:
od:

```

```
end proc:
```

```
> Rproc:=Compiler:-Compile(Rproc):
```

```
> ENO2:=proc(Y00::Matrix(datatype=float[8]),Phi0::Vector(datatype=
float[8]),F0::Matrix(datatype=float[8]),dt::float,N::integer,
M::integer,Nx::integer,v0::float)
local i::integer,j::integer,h::float[8],nx::float[8],vel::float
[8],vx::float[8],vy::float[8],phix::float[8],phiy::float[8],
phiave::float[8],jj::integer,phixb::float[8],phixf::float[8],
phixb2::float[8],phixf2::float[8],vxb::float[8],phiyb::float[8],
phiyf::float[8],vxf::float[8],phiyb2::float[8],phiyf2::float[8],
vyb::float[8],vyf::float[8],uf::float[8],ub::float[8],vf::float
[8],vb::float[8],tt::float[8],vv0::float[8],sd::float[8],
sdf::float[8],sdb::float[8],sdx::float[8],sdy::float[8],
sdxb::float[8],s1x::float[8],sdxf::float[8],sdyb::float[8],
sdyf::float[8],s1y::float[8],vx1::float[8],vx2::float[8],
vy1::float[8],vy2::float[8],alpha::float[8],beta::float[8];

h:=1.0/(N-2):
for i from 1 to N do
Y00[i,1]:=Y00[i,2]:
Y00[i,M]:=Y00[i,M-1]:
od:
for j from 1 to M do
Y00[1,j]:=Y00[2,j]:
Y00[N,j]:=Y00[N-1,j]:
od:
for i from 2 to N-1 do for j from 2 to M-1 do
vx:=0.0:vy:=0.0:phix:=0.0:phiy:=0.0:

sdx:=(Y00[i+1,j]-2*Y00[i,j]+Y00[i-1,j])/h:sdy:=(Y00[i,j+1]-2*Y00
[i,j]+Y00[i,j-1])/h:
vxb:=0:vxf:=0:vyb:=0:vyf:=0:
if i = 2 then sdxb:=(Y00[i,j]-2*Y00[i-1,j]+Y00[i-1,j])/h:else
sdxb:=(Y00[i,j]-2*Y00[i-1,j]+Y00[i-2,j])/h:end:
if sdx*sdxb>=0 then s1x:=1.0 else s1x:=0.0:end:
vx1:=(Y00[i,j]-Y00[i-1,j])/h+0.5*signum(sdx)*s1x*min(abs(sdx),abs
(sdxb)):

if i = N-1 then sdxf:=(Y00[i+1,j]-2*Y00[i+1,j]+Y00[i,j])/h:else
```

```

sdxf:=(Y00[i+2,j]-2*Y00[i+1,j]+Y00[i,j])/h:end:
if sdx*sdxf>=0 then slx:=1.0 else slx:=0.0:end:
vx2:=(Y00[i+1,j]-Y00[i,j])/h-0.5*signum(sdx)*slx*min(abs(sdx),abs
(sdxf)):

```

```

if j = 2 then sdyb:=(Y00[i,j]-2*Y00[i,j-1]+Y00[i,j-1])/h:else
sdyb:=(Y00[i,j]-2*Y00[i,j-1]+Y00[i,j-2])/h:end:
if sdy*sdyb>=0 then sly:=1.0 else sly:=0.0:end:
vy1:=(Y00[i,j]-Y00[i,j-1])/h+0.5*signum(sdy)*sly*min(abs(sdy),abs
(sdyb)):

```

```

if j = M-1 then sdyf:=(Y00[i,j+1]-2*Y00[i,j+1]+Y00[i,j])/h:else
sdyf:=(Y00[i,j+2]-2*Y00[i,j+1]+Y00[i,j])/h:end:
if sdy*sdyf>=0 then sly:=1.0 else sly:=0.0:end:
vy2:=(Y00[i,j+1]-Y00[i,j])/h+0.5*signum(sdy)*sly*min(abs(sdy),abs
(sdyf)):

```

```

if v0>=0 then
vx1:=max(vx1,0):vx2:=-min(vx2,0): else
vx1:=-min(vx1,0):vx2:=max(vx2,0): end:

```

```

if v0>=0 then
vy1:=max(vy1,0):vy2:=-min(vy2,0): else
vy1:=-min(vy1,0):vy2:=max(vy2,0): end:
nx:=sqrt(max(vx1,vx2)^2+max(vy1,vy2)^2):

```

```

F0[i,j]:=nx:
od:od:

```

```

end proc:

```

```

> WENO3:=proc(Y00::Matrix(datatype=float[8]),Phi0::Vector(datatype=
float[8]),F0::Matrix(datatype=float[8]),dt::float,N::integer,
M::integer,Nx::integer,v0::float)
local i::integer,j::integer,h::float[8],nx::float[8],vel::float
[8],vx::float[8],vy::float[8],phix::float[8],phiy::float[8],
phiave::float[8],jj::integer,phixb::float[8],phixf::float[8],
phixb2::float[8],phixf2::float[8],vxb::float[8],phiyb::float[8],
phiyf::float[8],vxf::float[8],phiyb2::float[8],phiyf2::float[8],
vyb::float[8],vyf::float[8],uf::float[8],ub::float[8],vf::float
[8],vb::float[8],tt::float[8],vv0::float[8],sd::float[8],

```

```
sdf::float[8],sdb::float[8],sdx::float[8],sdy::float[8],
sdxb::float[8],s1x::float[8],sdx::float[8],sdyb::float[8],
sdyf::float[8],s1y::float[8],vx1::float[8],vx2::float[8],
vy1::float[8],vy2::float[8],w1::float[8],w2::float[8],r1::float
[8],r2::float[8],alpha::float[8],beta::float[8],e1::float[8];
e1:=1e-6:
```

```
h:=1.0/(N-2):
```

```
vv0:=0.1:
```

```
for i from 1 to N do
```

```
Y00[i,1]:=Y00[i,2]:
```

```
Y00[i,M]:=Y00[i,M-1]:
```

```
od:
```

```
for j from 1 to M do
```

```
Y00[1,j]:=Y00[2,j]:
```

```
Y00[N,j]:=Y00[N-1,j]:
```

```
od:
```

```
for i from 2 to N-1 do for j from 2 to M-1 do
```

```
vx:=0.0:vy:=0.0:phix:=0.0:phiy:=0.0:
```

```
phix:=(Y00[i+1,j]-Y00[i-1,j])/2/h:
```

```
phiy:=(Y00[i,j+1]-Y00[i,j-1])/2/h:
```

```
if i = 2 then sdb:=Y00[i,j]-2*Y00[i-1,j]+Y00[i-1,j]: else sdb:=
Y00[i,j]-2*Y00[i-1,j]+Y00[i-2,j]:end:
```

```
sd:=Y00[i+1,j]-2*Y00[i,j]+Y00[i-1,j]:
```

```
if i = N-1 then sdf:=Y00[i,j]-2*Y00[i+1,j]+Y00[i+1,j]: else sdf:=
Y00[i+2,j]-2*Y00[i+1,j]+Y00[i,j]:end:
```

```
r1:=(e1+sdb^2)/(e1+sd^2):w1:=1/(1+2*r1^2):r2:=(e1+sdf^2)/(e1+
sd^2):w2:=1/(1+2*r2^2):
```

```
vx1:=phix-0.5*w1/h*(sd-sdb):
```

```
vx2:=phix-0.5*w2/h*(sdf-sd):
```

```
if j = 2 then sdb:=Y00[i,j]-2*Y00[i,j-1]+Y00[i,j-1]:else sdb:=Y00
[i,j]-2*Y00[i,j-1]+Y00[i,j-2]:end:
```

```
sd:=Y00[i,j+1]-2*Y00[i,j]+Y00[i,j-1]:
```

```
if j = M-1 then sdf:=Y00[i,j]-2*Y00[i,j+1]+Y00[i,j+1]: else sdf:=
Y00[i,j+2]-2*Y00[i,j+1]+Y00[i,j]:end:
```

```
r1:=(e1+sdb^2)/(e1+sd^2):w1:=1/(1+2*r1^2):r2:=(e1+sdf^2)/(e1+
sd^2):w2:=1/(1+2*r2^2):
```

```
vy1:=phiy-0.5*w1/h*(sd-sdb):
```

```

vy2:=phiy-0.5*w2/h*(sdf-sd):
alpha:=1.0:beta:=1.0:
if v0>=0 then
vx1:=max(vx1,0):vx2:=-min(vx2,0): else
vx1:=-min(vx1,0):vx2:=max(vx2,0): end:

if v0>=0 then
vy1:=max(vy1,0):vy2:=-min(vy2,0): else
vy1:=-min(vy1,0):vy2:=max(vy2,0): end:
nx:=sqrt(max(vx1,vx2)^2+max(vy1,vy2)^2):
F0[i,j]:=nx:
od:od:

```

```
end proc:
```

```

> UpW:=proc(Y00::Matrix(datatype=float[8]),Phi0::Vector(datatype=
float[8]),F0::Matrix(datatype=float[8]),dt::float,N::integer,
M::integer,Nx::integer,v0::float)
local i::integer,j::integer,h::float[8],nx::float[8],vel::float
[8],vx::float[8],vy::float[8],phix::float[8],phiy::float[8],
phiave::float[8],jj::integer,phixb::float[8],phixf::float[8],
phixb2::float[8],phixf2::float[8],vxb::float[8],phiyb::float[8],
phiyf::float[8],vxf::float[8],phiyb2::float[8],phiyf2::float[8],
vyb::float[8],vyf::float[8],uf::float[8],ub::float[8],vf::float
[8],vb::float[8],tt::float[8],vv0::float[8],sd::float[8],
sdf::float[8],sdb::float[8],sdx::float[8],sdy::float[8],
sdxb::float[8],s1x::float[8],sdxs::float[8],sdyb::float[8],
sdyf::float[8],s1y::float[8],vx1::float[8],vx2::float[8],
vy1::float[8],vy2::float[8],e1::float[8],w1::float[8],w2::float
[8],r1::float[8],r2::float[8],alpha::float[8],beta::float[8];
e1:=1e-15:

```

```
h:=1.0/(N-2):
```

```

vv0:=0.1:
for i from 1 to N do
Y00[i,1]:=Y00[i,2]:
Y00[i,M]:=Y00[i,M-1]:
od:
for j from 1 to M do
Y00[1,j]:=Y00[2,j]:
Y00[N,j]:=Y00[N-1,j]:
od:

```

```
for i from 2 to N-1 do for j from 2 to M-1 do
vx:=0.0:vy:=0.0:phix:=0.0:phiy:=0.0:
```

```
vx1:=(Y00[i,j]-Y00[i-1,j])/h:
vx2:=(Y00[i+1,j]-Y00[i,j])/h:
```

```
vy1:=(Y00[i,j]-Y00[i,j-1])/h:
vy2:=(Y00[i,j+1]-Y00[i,j])/h:
```

```
if v0>=0 then
vx1:=max(vx1,0):vx2:=-min(vx2,0): else
vx1:=-min(vx1,0):vx2:=max(vx2,0): end:
```

```
if v0>=0 then
vy1:=max(vy1,0):vy2:=-min(vy2,0): else
vy1:=-min(vy1,0):vy2:=max(vy2,0): end:
nx:=sqrt(max(vx1,vx2)^2+max(vy1,vy2)^2):
```

```
F0[i,j]:=nx:
od:od:
```

```
end proc:
```

```
> ENO2:=Compiler:-Compile(ENO2):
```

```
> WENO3:=Compiler:-Compile(WENO3):
```

```
> UpW:=Compiler:-Compile(UpW):
```

```
> printlevel:=1:
```

```
> pFactor := define_external("hw_PardisoFactor", ':-MAPLE', ':-LIB'
= "linalg"):
```

```
pSolve := define_external("hw_PardisoSolve", ':-MAPLE', ':-LIB' =
"linalg"):
```

```
pCSF := define_external("hw_SpCompressedSparseForm", ':-MAPLE',
':-LIB' = "linalg"):
```

```
> #NN:=20;
```

```
> ss:=proc(NN,delPhi,nn,tf,beta,e1)
```

```
local tt,V,V2,V1,TT,dt,vv0,ii,nt,Nt,N,h,w,Nx,MM,M,Ny,ntot,Ntot,
Nsp,Y00,Phi0,F0,ff,db,CB,CB0,Ymid,R0,x0,handle,Phiave;
```

```
N:=NN+2:h:=1.0/NN:w:=h/2:Nx:=NN:
```

```
> MM:=NN:M:=MM+2:Ny:=MM:ntot:=N*(M):Ntot:=ntot:Nsp:=5*(N-2)^2+8*
```

```

(N-1) :
> Y00:=Matrix(1..N,1..M,datatype=float[8]):F0:=Matrix(1..N,1..M,
datatype=float[8]):Phi0:=Vector(1..Ntot,datatype=float[8]):ff:=
copy(Phi0):db:=copy(ff):CB0:=Vector(N*M+1,datatype=integer[4])
:Ymid:=Matrix(1..N,1..M,datatype=float[8]):
> y0proc(NN,Y00,beta,e1):
> R0:=Vector(Nsp,datatype=integer[4]):x0:=Vector(Nsp,datatype=float
[8]):
> CBproc(N,M,CB0):
> CB0:=convert(CB0,Vector,datatype=integer[8]):
> Rproc(N,M,R0):
> R0:=convert(R0,Vector,datatype=integer[8]):
> evalf(Eqs11(N,M,Y00,F0,Phi0,evalf(0.1),ff));
> XX(N,M,Y00,Phi0,0.1,x0);handle := pFactor(CB0, R0, x0, 11)
:#handle := pFactor(CB, R, X, 11):
pSolve(ff,db,handle):
> PhiAdd(Ntot,Phi0,db):
> V2[0]:=(Phi0[ntot-2*N+N/2]/2+Phi0[ntot-2*N+N/2+1]/2)+h/2*0.1;
V1[0]:=Phi0[2*N];
V[0]:=Phi0[ntot-N]+0.5*h*0.1;
> TT[0]:=0;tt:=0;
vv0:=max(abs(Phi0[ntot-2*N+1]),abs(Phi0[ntot-2*N+N/2]),abs(Phi0
[ntot-2*N+N/2+1]),abs(Phi0[ntot-N]))):
> dt:=min(h/vv0/nn,tf-tt);
> #phiaveAdd(Y00,N,M,Ny);
> Phiave[0]:=phiaveAdd(Y00,N,M,Ny);

> Nt:=round(tf/dt);
#dt:=tf/Nt;

> #Ydat:=Array(1..Nt+1,1..N,1..M,datatype=float[8]):
> #YdatStore(Y00,Ydat,N,M,1);
> ii:=0:TT[0];tt;
> while tt<tf do
delPhi(Y00,Phi0,F0,evalf(dt),N,M,Nx,0.1):
EFAdd(Y00,Ymid,Phi0,F0,dt,N,M,e1);
Eqs11(N,M,Ymid,F0,Phi0,0.1,ff);
XX(N,M,Ymid,Phi0,0.1,x0);handle := pFactor(CB0, R0, x0, 11):
pSolve(ff,db,handle):
PhiAdd(Ntot,Phi0,db):
delPhi(Ymid,Phi0,F0,evalf(dt),N,M,Nx,0.1):
EFAdd2(Y00,Ymid,Phi0,F0,dt,N,M,e1);

```

```

Eqs11(N,M,Ymid,F0,Phi0,evalf(0.1),ff);
XX(N,M,Ymid,Phi0,0.1,x0);handle := pFactor(CB0, R0, x0, 11)
:#handle := pFactor(CB, R, X, 11):
pSolve(ff,db,handle):
PhiAdd(Ntot,Phi0,db):
delPhi(Ymid,Phi0,F0,evalf(dt),N,M,Nx,0.1):
EFAdd3(Y00,Ymid,Phi0,F0,dt,N,M,e1);
Eqs11(N,M,Y00,F0,Phi0,evalf(0.1),ff);
XX(N,M,Y00,Phi0,0.1,x0);handle := pFactor(CB0, R0, x0, 11)
:#handle := pFactor(CB, R, X, 11):
pSolve(ff,db,handle):
PhiAdd(Ntot,Phi0,db):
ii:=ii+1:#print(i);
V2[ii]:=(Phi0[ntot-2*N+N/2]/2+Phi0[ntot-2*N+N/2+1]/2)+h/2*0.1;
#print(ii,V[ii]);
#V1[ii]:=Phi0[ntot]+h/2*0.1;
V1[ii]:=Phi0[2*N];
V[ii]:=Phi0[ntot-N]+0.5*h*0.1;
TT[ii]:=TT[ii-1]+dt:tt:=tt+dt:
#YdatStore(Y00,Ydat,N,M,ii+1);
#Phiave[ii]:=phiaveAdd(Y00,N,M,Ny);
vv0:=max(abs(Phi0[ntot-2*N+1]),abs(Phi0[ntot-2*N+N/2]),abs(Phi0
[ntot-2*N+N/2+1]),abs(Phi0[ntot-N]))):
#dt:=h/vv0/nn;
dt:=min(h/vv0/nn,tf-tt);#print(tt,ii,dt);
gc();
end:

> V2[ii];
>
end proc:
#time[real]()-t11;time()-t12;NN;
> ssapp:=proc(NN,delPhi,nn,tf,beta,e1)
local tt,V,V2,V1,TT,dt,vv0,ii,nt,Nt,N,h,w,Nx,MM,M,Ny,ntot,Ntot,
Nsp,Y00,Phi0,F0,ff,db,CB,CB0,Ymid,R0,x0,handle,Phiave,Phimid,
Phitemp;
N:=NN+2:h:=1.0/NN:w:=h/2:Nx:=NN:
> MM:=NN:M:=MM+2:Ny:=MM:ntot:=N*(M):Ntot:=ntot:Nsp:=5*(N-2)^2+8*
(N-1):
> Y00:=Matrix(1..N,1..M,datatype=float[8]):F0:=Matrix(1..N,1..M,
datatype=float[8]):Phi0:=Vector(1..Ntot,datatype=float[8]):ff:=
copy(Phi0):db:=copy(ff):CB0:=Vector(N*M+1,datatype=integer[4])

```

```

:Ymid:=Matrix(1..N,1..M,datatype=float[8]):Phitemp:=Vector(1..
Ntot,datatype=float[8]):Phimid:=Vector(1..Ntot,datatype=float[8])
:
> y0proc(NN,Y00,beta,e1):
> R0:=Vector(Nsp,datatype=integer[4]):x0:=Vector(Nsp,datatype=float
[8]):
> CBproc(N,M,CB0):
> CB0:=convert(CB0,Vector,datatype=integer[8]):
> Rproc(N,M,R0):
> R0:=convert(R0,Vector,datatype=integer[8]):
> evalf(Eqs11(N,M,Y00,F0,Phi0,evalf(0.1),ff));
> XX(N,M,Y00,Phi0,0.1,x0);handle := pFactor(CB0, R0, x0, 11)
:#handle := pFactor(CB, R, X, 11):
pSolve(ff,db,handle):
> PhiAdd(Ntot,Phi0,db):
> V2[0]:=(Phi0[ntot-2*N+N/2]/2+Phi0[ntot-2*N+N/2+1]/2)+h/2*0.1;
V1[0]:=Phi0[2*N];
V[0]:=Phi0[ntot-N]+0.5*h*0.1;
> TT[0]:=0;tt:=0;
vv0:=max(abs(Phi0[ntot-2*N+1]),abs(Phi0[ntot-2*N+N/2]),abs(Phi0
[ntot-2*N+N/2+1]),abs(Phi0[ntot-N])):
> dt:=h/vv0/nn;
> #phiaveAdd(Y00,N,M,Ny);
> Phiave[0]:=phiaveAdd(Y00,N,M,Ny);

> Nt:=round(tf/dt);
#dt:=tf/Nt;

> #Ydat:=Array(1..Nt+1,1..N,1..M,datatype=float[8]):
> #YdatStore(Y00,Ydat,N,M,1);
> ii:=0:TT[0];tt;
> while tt<tf do
PhiSwitch(Ntot,Phi0,Phitemp):
delPhi(Y00,Phi0,F0,evalf(dt),N,M,Nx,0.1):
EFAdd(Y00,Ymid,Phi0,F0,dt,N,M,e1);
Eqs11(N,M,Ymid,F0,Phi0,0.1,ff);
XX(N,M,Ymid,Phi0,0.1,x0);handle := pFactor(CB0, R0, x0, 11):
pSolve(ff,db,handle):
PhiAdd(Ntot,Phi0,db):
MidPred(Ntot,Phi0,Phitemp,Phimid):
delPhi(Ymid,Phi0,F0,evalf(dt),N,M,Nx,0.1):
EFAdd2(Y00,Ymid,Phi0,F0,dt,N,M,e1);

```

```

delPhi (Ymid,Phimid,F0,evalf(dt),N,M,Nx,0.1):
EFAdd3 (Y00,Ymid,Phimid,F0,dt,N,M,e1);
ii:=ii+1:#print(i);
V2[ii]:=(Phi0[ntot-2*N+N/2]/2+Phi0[ntot-2*N+N/2+1]/2)+h/2*0.1;
#print(ii,V[ii]);
#V1[ii]:=Phi0[ntot]+h/2*0.1;
V1[ii]:=Phi0[2*N];
V[ii]:=Phi0[ntot-N]+0.5*h*0.1;
TT[ii]:=TT[ii-1]+dt:tt:=tt+dt:
#YdatStore(Y00,Ydat,N,M,ii+1);
#Phiave[ii]:=phiaveAdd(Y00,N,M,Ny);
vv0:=max(abs(Phi0[ntot-2*N+1]),abs(Phi0[ntot-2*N+N/2]),abs(Phi0
[ntot-2*N+N/2+1]),abs(Phi0[ntot-N])):
#dt:=h/vv0/nn;
dt:=min(h/vv0,tf-tt);#print(tt,ii,dt);
gc();
end:
V2[ii];
end proc:
#time[real]()-t11;time()-t12;NN;

```

A procedure is written to find the value of the potential at $t=tf$ using different upwind methods using both SSP RK3 and SSP RK3a. $Nt1$ can be set to 8 to get results till 1280.

```
> Nt1:=5;
```

```
Nt1 := 5
```

(4)

```

> for i from 1 to Nt1 do
ntot:=10*2^(i-1):
t11:=time[real]():t12:=time():ss1[i]:=CodeTools:-Usage(ss(ntot,
WENO3,1,2.0,0.5,1e-9)):
if i=1 then err:=1: else err:=ss1[i]-ss1[i-1]:end:
print(time()-t12,time[real]()-t11,ntot,err,ss1[i]):od:

```

```
memory used=114.08KiB, alloc change=0 bytes, cpu time=3.20s, real time=408.00ms,
gc time=3.20s
```

```
3.203, 0.415, 10, 1, 0.168213468716081
```

```
memory used=202.02KiB, alloc change=0 bytes, cpu time=7.03s, real time=898.00ms,
gc time=7.03s
```

```
7.031, 0.898, 20, 0.00101555384254789, 0.169229022558629
```

```
memory used=464.83KiB, alloc change=0 bytes, cpu time=14.06s, real time=1.79s,
gc time=11.94s
```

```
14.063, 1.787, 40, 0.000693040364631720, 0.169922062923261
```

```
memory used=1.33MiB, alloc change=0 bytes, cpu time=37.95s, real time=4.82s, gc
```

time=23.59s

37.953, 4.824, 80, 0.000248696712060820, 0.170170759635322

memory used=4.56MiB, alloc change=0 bytes, cpu time=2.81m, real time=21.29s, gc time=45.47s

168.312, 21.291, 160, 0.0000812191499713066, 0.170251978785293

(5)

> for i from 1 to Nt1 do

 ntot:=10*2^(i-1):

 t11:=time[real]():t12:=time():ss1[i]:=CodeTools:-Usage(ss(ntot,
 ENO2,1,2.0,0.5,1e-9)):

 if i=1 then err:=1: else err:=ss1[i]-ss1[i-1]:end:

 print(time()-t12,time[real]()-t11,ntot,err,ss1[i]):od:

memory used=114.48KiB, alloc change=0 bytes, cpu time=3.33s, real time=442.00ms,
gc time=3.08s

3.328, 0.442, 10, 1, 0.167167421328536

memory used=202.77KiB, alloc change=0 bytes, cpu time=6.66s, real time=846.00ms,
gc time=6.42s

6.656, 0.846, 20, 0.00157648593052276, 0.168743907259059

memory used=465.79KiB, alloc change=0 bytes, cpu time=15.02s, real time=1.93s,
gc time=12.64s

15.016, 1.927, 40, 0.000985188800596504, 0.169729096059655

memory used=1.34MiB, alloc change=0 bytes, cpu time=38.17s, real time=4.84s, gc
time=23.05s

38.172, 4.843, 80, 0.000360932005780162, 0.170090028065435

memory used=4.56MiB, alloc change=0 bytes, cpu time=2.91m, real time=22.01s, gc
time=47.73s

174.735, 22.006, 160, 0.000125191312049688, 0.170215219377485

(6)

> for i from 1 to Nt1 do

 ntot:=10*2^(i-1):

 t11:=time[real]():t12:=time():ss1[i]:=CodeTools:-Usage(ss(ntot,
 UpW,1,2.0,0.5,1e-9)):

 if i=1 then err:=1: else err:=ss1[i]-ss1[i-1]:end:

 print(time()-t12,time[real]()-t11,ntot,err,ss1[i]):od:

memory used=114.42KiB, alloc change=0 bytes, cpu time=3.38s, real time=434.00ms,
gc time=3.12s

3.375, 0.434, 10, 1, 0.166527181907120

memory used=203.07KiB, alloc change=0 bytes, cpu time=6.95s, real time=891.00ms,
gc time=6.33s

6.953, 0.891, 20, 0.0000564615675806568, 0.166583643474701

memory used=464.99KiB, alloc change=0 bytes, cpu time=14.55s, real time=1.87s, gc time=12.30s

14.547, 1.869, 40, 0.00150597947611711, 0.168089622950818

memory used=1.33MiB, alloc change=0 bytes, cpu time=38.12s, real time=4.87s, gc time=23.77s

38.125, 4.866, 80, 0.00102702187608120, 0.169116644826899

memory used=4.56MiB, alloc change=0 bytes, cpu time=2.81m, real time=21.23s, gc time=47.78s

168.391, 21.234, 160, 0.000551747971749089, 0.169668392798648

(7)

> Nt1:=5;

Nt1 := 5

(8)

> for i from 1 to Nt1 do

ntot:=10*2^(i-1):

t11:=time[real]():t12:=time():ss1[i]:=CodeTools:-Usage(ssapp
(ntot,WENO3,1,2.0,0.5,1e-9)):

if i=1 then err:=1: else err:=ss1[i]-ss1[i-1]:end:

print(time()-t12,time[real]()-t11,ntot,err,ss1[i]):od:

memory used=118.80KiB, alloc change=0 bytes, cpu time=3.25s, real time=431.00ms, gc time=3.25s

3.250, 0.431, 10, 1, 0.168071391535308

memory used=209.43KiB, alloc change=0 bytes, cpu time=6.67s, real time=875.00ms, gc time=5.94s

6.672, 0.875, 20, 0.00116185057162599, 0.169233242106934

memory used=486.76KiB, alloc change=0 bytes, cpu time=12.11s, real time=1.56s, gc time=11.11s

12.109, 1.555, 40, 0.000703726027799584, 0.169936968134733

memory used=1.42MiB, alloc change=0 bytes, cpu time=29.44s, real time=3.79s, gc time=23.20s

29.438, 3.789, 80, 0.000245027198407438, 0.170181995333141

memory used=4.92MiB, alloc change=0 bytes, cpu time=92.78s, real time=11.78s, gc time=47.47s

92.781, 11.782, 160, 0.0000772240456361206, 0.170259219378777

(9)

> for i from 1 to Nt1 do

ntot:=10*2^(i-1):

t11:=time[real]():t12:=time():ss1[i]:=CodeTools:-Usage(ssapp
(ntot,ENO2,1,2.0,0.5,1e-9)):

```
if i=1 then err:=1: else err:=ss1[i]-ss1[i-1]:end:  
print(time()-t12,time[real]()-t11,ntot,err,ss1[i]):od:
```

```
memory used=119.35KiB, alloc change=0 bytes, cpu time=3.14s, real time=  
398.00ms, gc time=3.14s
```

```
3.140, 0.398, 10, 1, 0.167130290323496
```

```
memory used=209.79KiB, alloc change=0 bytes, cpu time=6.28s, real time=  
803.00ms, gc time=6.16s
```

```
6.281, 0.803, 20, 0.00163245508331705, 0.168762745406813
```

```
memory used=487.45KiB, alloc change=0 bytes, cpu time=12.97s, real time=1.66s,  
gc time=12.34s
```

```
12.969, 1.658, 40, 0.000983339821275159, 0.169746085228089
```

```
memory used=1.42MiB, alloc change=0 bytes, cpu time=29.38s, real time=3.73s, gc  
time=23.25s
```

```
29.375, 3.733, 80, 0.000354563359009324, 0.170100648587098
```

```
memory used=4.93MiB, alloc change=0 bytes, cpu time=95.58s, real time=12.04s,  
gc time=46.34s
```

```
95.578, 12.039, 160, 0.000120502967830033, 0.170221151554928
```

(10)

```
> for i from 1 to Nt1 do  
ntot:=10*2^(i-1):  
t11:=time[real]():t12:=time():ss1[i]:=CodeTools:-Usage(ssapp  
(ntot,UpW,1,2.0,0.5,1e-9)):  
if i=1 then err:=1: else err:=ss1[i]-ss1[i-1]:end:  
print(time()-t12,time[real]()-t11,ntot,err,ss1[i]):od:
```

```
memory used=119.35KiB, alloc change=0 bytes, cpu time=3.25s, real time=  
425.00ms, gc time=3.00s
```

```
3.250, 0.425, 10, 1, 0.166295940733517
```

```
memory used=210.17KiB, alloc change=0 bytes, cpu time=6.66s, real time=  
844.00ms, gc time=6.53s
```

```
6.657, 0.844, 20, 0.000291903071384103, 0.166587843804901
```

```
memory used=486.81KiB, alloc change=0 bytes, cpu time=13.08s, real time=1.65s,  
gc time=12.20s
```

```
13.078, 1.649, 40, 0.00150384918732160, 0.168091692992222
```

```
memory used=1.42MiB, alloc change=0 bytes, cpu time=29.41s, real time=3.74s, gc  
time=23.42s
```

```
29.406, 3.744, 80, 0.00102690121988763, 0.169118594212110
```

```
memory used=4.92MiB, alloc change=0 bytes, cpu time=91.64s, real time=11.62s,
```

gc time=46.61s

91.641, 11.621, 160, 0.000551245843922299, 0.169669840056032

(11)