Supplementary material to the manuscript Relaxation in Scalar Gravitational Field Theory

Riccardo Fantoni[1]

Università di Trieste, Dipartimento di Fisica, strada Costiera 11, 34151 Grignano (Trieste), Italy

We give the listing of the characteristic approximation FORTRAN code used for the numerical results presented in the manuscript "Relaxation in Scalar Gravitational Field Theory" appeared in the journal Gravitation & Cosmology Issue 2 year 2026.

1 The code

This is the code used for the exact numerical integration.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                   1 SHELL CLUSTER
c dimt = number of timesteps in the integration
c dimg = dimension of the uniform r grid
c
c INPUT r0=shell radius
c       mr=shell rest mass
c       xi=up/up(circular)
c       dt=time-step
c       rot=dr/dt       dr=grid spacing
c OUTPUT
c       fort.8 = (t,rp)
c       fort.9 = (rp,utt,utlr)
c       erp=equilibrium radius
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit none
      include  'cluster.p'
c INPUT
      real*8  r0,mr,xi
      real*8  dt
      integer rot
c OUTPUT
      real*8  erp,rp,utt,utlr
c INTERNAL
      real*8  phirp,e2p,fpl,fpr,phiprp,e2ppr,st
      real*8  a,ea,dr
      real*8  ut,up,utlp,am2
      real*8  xx(0:imax),yy(-imax:imax)
      real*8  rg(-imax:imax),phi(0:imax)
      integer i,tsteps,jp,dimg,dimt
      parameter(dimg=1000)
      parameter(dimt=60000)
c =======================INPUT DATA============================
      call in(rp,mr,xi,dr,rot,dt)
      r0=rp
c ====================INITIAL CONDITION=======================
      tsteps=0
c -------------uniform grid in r (spacing dr)-------------------
      do i=-dimg,dimg
         rg(i)=dble(i)*dr
      enddo
c ----------particle-----------------------------------------------
c tangential orbit (utlr=0)
      utlr=0.d0
c find angular velocity for the circular orbit at rp
      call phi1(mr,rp,a,up)
c angular momentum for circular orbits (in a time
c independent field) is:
c utlp(circ)=ulp(circ)*exp(phi)=up(circ)*r*r*exp(phi)
c set utlp=xi*utlp(circ) = constant of motion
      utlp=xi*rp**2*exp(a/rp)*up
      am2 =utlp**2
      up=utlp/(rp**2*exp(a/rp))
      ut=sqrt(1.d0+(rp*up)**2)
c initial source term
      st=exp(a/rp)*mr/(2.d0*rp*ut)
c ----------field---------------------------------------------------
c xx(r,0)=yy(r,0)=(r*phi(r,0)),r
c phi(r,0)=a/rp    r <= rp
c phi(r,0)=a/r     r >  rp
      jp=nint(rp/dr)

c real space       r >= 0
      do i=0,jp-1
         xx(i)=.5d0*a/rp
         yy(i)=.5d0*a/rp
         phi(i)=a/rp
      enddo
      do i=jp,dimg
         xx(i)=0.d0
         yy(i)=0.d0
         phi(i)=a/rg(i)
      enddo
c immaginary space r < 0
      do i=-dimg,-jp-1
         yy(i)=0.d0
      enddo
      do i=-jp,-1
         yy(i)=.5d0*a/rp
      enddo
c ====================NEXT TIMESTEP==========================
 100  tsteps=tsteps+1
      if(mod(tsteps,rot).ne.0) goto 15

c ----------evolve field-----------------------------------------
c reinterpolate phi(rp) to find new source term
      jp   =nint(rp/dr)
      phirp=phi(jp)
      e2p  = exp(2.d0*phirp)
      utt  = sqrt(e2p+utlr**2+(utlp/rp)**2)
c the new source term is
      st   = .5d0*e2p*mr/(rp*utt)
c  evolve the field
      call evphi(dimg,dr,rg,st,rp,xx,yy,phi)

c ----------evolve particle-------------------------------------
c find e2p=exp(2*phi(rp))
 15   jp    = nint(rp/dr)
      phirp = phi(jp)
      e2p   = exp(2.d0*phirp)
c find e2ppr=e2p*(phi,r(rp-)+phi,r(rp+))/2
      fpl   = (xx(jp-1)+yy(jp-1)-phirp)/rp
      fpr   = (xx(jp+1)+yy(jp+1)-phirp)/rp
      phiprp= (fpl+fpr)*.5d0
      e2ppr = e2p*phiprp
c evolve the particle with 4-th order Runge-Kutta
      call runge4(am2,e2p,e2ppr,dt,rp,utlr,rp,utlr)

c write fort.8 :[t,r(t)] and fort.9 :[t,utt(t),utlr(t)]
      write(8,*) tsteps*dt,rp,phirp
      write(9,*) rp,utt,utlr
      if(tsteps.eq.dimt) goto 200
      goto 100

c estimate the final equilibrium radius erp
 200  call eqrp1(utlp,mr,erp,ea)
c write output
      call out(mr,r0,erp,xi,dt,dr,dimt,dimg)

      stop
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  Read initial data
c INPUT
c  rp    = initial shell radius
c  mr    = shell rest mass
c  xi    = ratio up/up(circular)
c  dt    = time-step
c  rot   = dr/dt(>=1 Courant stability condition)
c
```

[1]e-mail: riccardo.fantoni@scuola.istruzione.it

```
c OUTPUT(all above +)
c  dr    = grid spacing
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine in(rp,mr,xi,dr,rot,dt)
      implicit none
      real*8 rp,mr,xi,dr,dt
      integer rot

      write(*,*) 'initial radius rp'
      read(*,*) rp
      write(*,*) 'rest mass mr'
      read(*,*) mr
      write(*,*) 'ratio utlp/utlp(circular)'
      read(*,*) xi
      write(*,*) 'time-step dt'
      read(*,*) dt
      write(*,*) 'ratio dr/dt=[integer>=1]'
      read(*,*) rot
      dr=dt*dble(rot)

      return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  4-th order runge-kutta
c  advances to the next time step (h) the equations
c  dr/dt=f(r,u)
c  du/dt=g(r.u)
c  where u=utlr (u tilde-low-r)
c  f(r,u)=u/utu0
c  g(r,u)=utlp**2/(utu0*r**3)-exp(2*phi)*phi,r/utu0
c  utu0=sqrt(exp(2*phi)+u**2+(utlp/r)**2)
c  and phi   = potential at r,u
c      phi,r = d(phi)/dr at r,u
c INPUT  am2(=utlp**2 angular momentum squared),
c        e2p(=exp(2*phi)),e2ppr(=exp(2*phi)*phi,r),
c        h(=time step),ri,ui(=initial values for r,u)
c OUTPUT r,u  (=final   values for r,u)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine runge4(am2,e2p,e2ppr,h,ri,ui,r,u)
      implicit none
c INPUT
      real*8 am2,e2p,e2ppr,h,ri,ui
c OUTPUT
      real*8 u,r
c INTERNAL
      real*8 f,g,u0
      real*8 k1,k2,k3,k4,l1,l2,l3,l4
      real*8 k1o2,k2o2,l1o2,l2o2
      real*8 inv6
      parameter(inv6=1/6.d0)
      u0(r,u)= sqrt(e2p+u**2+am2/r**2)
      f(r,u) = u/u0(r,u)
      g(r,u) = am2/(u0(r,u)*r**3)-e2ppr/u0(r,u)

      k1  = h*f(ri,ui)
      l1  = h*g(ri,ui)
      k1o2= .5d0*k1
      l1o2= .5d0*l1
      k2  = h*f(ri+k1o2,ui+l1o2)
      l2  = h*g(ri+k1o2,ui+l1o2)
      k2o2= .5d0*k2
      l2o2= .5d0*l2
      k3  = h*f(ri+k2o2,ui+l2o2)
      l3  = h*g(ri+k2o2,ui+l2o2)
      k4  = h*f(ri+k3,ui+l3)
      l4  = h*g(ri+k3,ui+l3)

      r   = ri+inv6*(k1+2.d0*(k2+k3)+k4)
      u   = ui+inv6*(l1+2.d0*(l2+l3)+l4)

      return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  Integrate the 1D wave equation with a delta
c  function as the source
c      -(r*phi),tt+(r*phi),rr=2*st*delta(r-rp)
c  rewritten as
c      xx,t=xx,r+st*delta(r-rp)
c      yy,t=-yy,r-st*delta(r-rp)
c  where
c      xx=(v+w)/2
c      yy=(v-w)/2
c  and
c      v=(r*phi),r
c      w=(r*phi),t
c  add an image to ensure finiteness of phi(0,t) forall t
c      xx(0,t)=yy(0,t)  at all times
c
c INPUT  dr (= grid spacing), rg(-imax:imax) (= grid),
c        st (= source term),  rp (= shell radius),
c        xxo(0:imax), yyo(-imax:imax) (= old "field")
c
c OUTPUT xxo(0:imax), yyo(-imax:imax), phi(0:imax) (=field)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine evphi(dimg,dr,rg,st,rp,xxo,yyo,phi)
      implicit none
      include 'cluster.p'
c INPUT
      integer dimg
      real*8  st,dr,rp
      real*8  rg(-imax:imax)
c OUTPUT
      real*8  xxo(0:imax),yyo(-imax:imax)
      real*8  phi(0:imax)
c INTERNAL
      real*8  xx(0:imax),yy(-imax:imax)
      real*8  dt,psi
      integer i

c check for rp>=rg(dimg-1)
      if(rp.ge.rg(dimg-1)) then
         write(*,*) 'particle out of right grid margin !!!'
         stop
      endif
c field timestep
      dt=dr
c yy(r)=yyo(r-dt)+st*step[rp,rp+dt]-st*step[-rp,-rp+dt]
c xx(r)=xxo(r+dt)-st*step[rp-dt,rp]
      yy(-dimg)=0.d0
      do i=-dimg+1,-1
         yy(i)=yyo(i-1)
         if(-rp.le.rg(i).and.rg(i).lt.-rp+dt) then
            yy(i)=yy(i)-st
         endif
      enddo
      do i=0,dimg-1
         xx(i)=xxo(i+1)
         yy(i)=yyo(i-1)
         if(rp-dt.le.rg(i).and.rg(i).lt.rp) then
            xx(i)=xx(i)-st
         elseif(rp.le.rg(i).and.rg(i).lt.rp+dt) then
            yy(i)=yy(i)+st
         endif
      enddo
      xx(dimg)=0.d0
      yy(dimg)=yyo(dimg-1)
c rewrite xx and yy
      do i=-dimg,-1
         yyo(i)=yy(i)
      enddo
      do i=0,dimg
         xxo(i)=xx(i)
         yyo(i)=yy(i)
      enddo
c integrate x+y starting from the origin
c using trapezoidal method (order dr**3)
      psi=.5d0*(xx(0)+yy(0))
      do i=1,dimg
         psi=psi+xx(i)+yy(i)
c the gravitational potential is
         phi(i)=dr*(psi-.5d0*(xx(i)+yy(i)))/rg(i)
      enddo
      phi(0)=phi(1)
c check boundary condition at r=0
      if(xx(0).ne.yy(0)) then
         print *,'xx(0) <> yy(0) !!!!!!!!'
      endif

      return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Given rp (and ur=0) solves for a and up in:
c 1.0)   ut = sqrt(1+ur**2+(rp*up)**2)
c 1.1)   a  =-exp(a/rp)*mr/ut
c 2  )   up = sqrt(-a/(2*rp**3))
c rewritten as
c      -a = exp(a/rp)*mr/sqrt(1-a/(2*rp))
c
c INPUT   mr (= rest mass),rp (= shell radius)
c OUTPUT  a ("potential"), up (= angular velocity)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine phi1(mr,rp,a,up)
      implicit none
c INPUT
      real*8  mr,rp
c OUTPUT
      real*8  a,up
c INTERNAL
      real*8  ao,sqti,ep,f,fp
      real*8  acc
      parameter(acc=1.d-15)

      a=0.d0
c start the Newton iteration
 10   sqti=1.d0/sqrt(1.d0-a/(2.d0*rp))
      ep=exp(a/rp)
      f=a+mr*ep*sqti
      fp=1.d0+mr*ep*(sqti+.25d0*sqti**3)/rp
      ao=a
      a=ao-f/fp
```

```
      if(abs(f).gt.acc) goto 10

c the angular velocity rp*up**2=(a/rp**2)/2
      up=sqrt(-a/(2.d0*rp**3))
      return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c        Finds the equilibrium radius
c given utlp solves for a and rp in:
c 1.0)   utt    = sqrt(exp(2*a/rp)+(utlp/rp)**2)
c 1.1)   a      =-exp(2*a/rp)*mr/utt
c 2)     utlp**2=-exp(2*a/rp)*rp**3*(a/rp**2)/2
c rewritten as
c     y=4*a**4/(a**4-(2*utlp*mr)**2)
c 1) utlp**2*(y/a**2)=-exp(y)         -------> find a (<0)
c 2) r=(a**4-(2*utlp*mr)**2)/(2*a**3) -------> find r (>0)
c INPUT   utlp (= angular momentum), mr (= rest mass)
c OUTPUT  erp (= equil  radius),ea (=  equil "potential")
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine eqrp1(utlp,mr,erp,ea)
      implicit none
c INPUT
      real*8 mr,utlp
c OUTPUT
      real*8 ea,erp
c INTERNAL
      real*8 a4,y,ai,af,afo,fi,ff
      real*8 acc,u2,fu2m2
      parameter(acc=1.d-10)

      u2=utlp**2
      fu2m2=4.d0*u2*mr**2
c upper limit
      ai=0.d0
      fi=1
c find the lower limit
      af=-sqrt(2.d0*u2*(-1.d0+sqrt(1.d0+(-mr/utlp)**2)))
c start the secant iteration
 10   a4=af**4
      y =4.d0*a4/(a4-fu2m2)
      ff=u2*y/af**2+exp(y)
      afo=af
      af=afo-ff*(afo-ai)/(ff-fi)
      if(abs((af-afo)/afo).gt.acc) then
         ai=afo
         fi=ff
         goto 10
      endif
c found a find r
      erp=(af**4-fu2m2)/(2.d0*af**3)
      ea =af

      return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  writes parameters in output
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine out(mr,ri,rf,xi,dt,dr,dimt,dimg)
      implicit none
      real *8 mr,ri,rf,dt,dr,xi
      integer dimt,dimg

c rest mass
      write(*,*) 'mr=',mr
c initial radius
      write(*,*) 'ri=',ri
c xi=up/up(circular)
      write(*,*) 'xi=',xi
c final equilibrium radius
      write(*,*) 'rf=',rf
c time step
      write(*,*) 'dtf=',dt
c grid spacing
      write(*,*) 'dr=',dr
c total integration time=dimt*dtf
      write(*,*) 'dimt=',dimt
c grid dimension r in [0,dimg*dr]
      write(*,*) 'dimg=',dimg

      return
      end
```

This is the code used for enveloping the numerical integration.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c  Relaxation to virial equilibrium
c
c INPUT  r0=shell initial radius
c        mr=shell rest mass
c        xi=up/up(circular)
c
c OUTPUT fort.10 : r_max,utt(r_max),xi
c        fort.14 : time,r_max
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit none
c INPUT
      real*8  r0,mr,xi
c OUTPUT
      real*8  utt,r(0:1000),tt
c INTERNAL
      real*8  dr,dtdr(0:1000),etot,dedt,dedr
      real*8  a,ea,rf,utlp,up,am2
      real*8  e2p,npo2pi,xi2
      integer i,np
      parameter(np=999)
c ======================INPUT DATA==========================
      write(*,*) 'initial radius r0'
      read(*,*) r0
      write(*,*) 'rest mass mr'
      read(*,*) mr
      write(*,*) 'ratio utlp/utlp(circular)'
      read(*,*) xi
      if(xi.ge.sqrt(2.d0)) then
         print *,'qust.f uses Newtonian approx. : xi < sqrt(2)'
         stop
      endif
c ======================INITIALIZATION=======================
c find angular velocity for the circular orbit at r0
      call phi1(mr,r0,a,up)
      utlp =xi*r0**2*exp(a/r0)*up
      am2  =utlp**2
c find final equilibrium radius rf and particle energy
      call eqrp1(utlp,mr,rf,ea)
      dr   =(rf-r0)/dble(np)
      print *,'initial radius, potential=',r0,a/r0
      print *,'final   radius, potential=',rf,ea/rf
      print *,'initial energy utt=',sqrt(exp(2.d0*a/r0) +am2/r0**2)
      print *,'final   energy utt=',sqrt(exp(2.d0*ea/rf)+am2/rf**2)
c =================r_{max},xi,utt(r_{max})=====================
      do i=0,np-1
         r(i)=r0+dble(i)*dr
         call phi1(mr,r(i),a,up)
         e2p=exp(2.d0*a/r(i))
         utt=sqrt(e2p+am2/r(i)**2)
c the new xi at r(i) is
         xi=utlp/(exp(a/r(i))*up*r(i)**2)
c write fort.10 : (r,utt,xi)
         write(10,*) r(i),utt,xi
c the total energy is then
         etot=mr*utt
         xi2=xi**2
c calculate detot/dr
         dedr=(a*r(i)*(4.d0-7.d0*xi2)+a**2*2.d0*xi2+r(i)**2*4.d0
     $  *(xi2-2.d0))/(xi2*r(i)**3*(7.d0*a*r(i)-2.d0*a**2-
     $   4.d0*r(i)**2))
         dedr=mr*dedr*am2/utt
c calculate detot/dt
         npo2pi=sqrt(2.d0*r(i)**3/(a*(xi2-2.d0)**3))
         dedt=-(sqrt(2.d0)/9.d0)*mr**2*(sqrt(-a)**5/sqrt(r(i))**7)
     $        *((1.d0-xi2)**2/xi**7)*(5.d0-2.d0*xi2+xi**4)/npo2pi
c calculate dr/dt
         dtdr(i)=dedr/dedt
      enddo
c ====================t,r_max==================================
      write(14,*) 0,r0
c integrate (dt/dr) to get t(r)
      tt=.5d0*dtdr(1)
      do i=1,np-1
         tt=tt+dtdr(i)
c make graph (t(r),r)
         write(14,*) dr*(tt-.5d0*dtdr(i)),r(i)
      enddo

 30   stop
      end
```

This is the code used for the quasistatic integration.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c            1 SHELL QUASISTATIC CLUSTER
c
c INPUT rp=shell radius
c       mr=shell rest mass
c       xi=up/up(circular)
c       dt=integration timestep
c       time=simulation duration
c OUTPUT
c       fort.18 = (t,rp)
c       fort.19 = (rp,utt,utlr)
c       erp=equilibrium radius
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit none
c INPUT
      real*8  rp,mr,xi,dt,time
c OUTPUT
      real*8  utlr,utt,erp,ea
c INTERNAL
      real*8  am2,e2p,e2ppr
      real*8  up,a,utlp
      integer tsteps,dimt
c ========================INPUT DATA============================
      write(*,*) 'initial radius rp'
      read(*,*) rp
      write(*,*) 'rest mass mr'
      read(*,*) mr
      write(*,*) 'ratio utlp/utlp(circular)'
      read(*,*) xi
      write(*,*) 'time step dt'
      read(*,*) dt
      write(*,*) 'time lenght'
      read(*,*) time
      dimt=int(time/dt)
c ====================INITIAL CONDITION=======================
      tsteps=0
c ----------particle-----------------------------------
c tangential orbit (utlr=0)
      utlr=0.d0
c find angular velocity for the circular orbit at rp
      call phi1(mr,rp,a,up)
c angular momentum for circular orbits (in a time
c independent field) is:
c utlp(circ)=ulp(circ)*exp(phi)=up(circ)*r*r*exp(phi)
c set utlp=xi*utlp(circ) = constant of motion
      utlp=xi*rp**2*exp(a/rp)*up
      am2 =utlp**2
c ----------field--------------------------------------
c phi(r)=a/rp for r<=rp
c phi(r)=a/r  for r> rp
c ====================NEXT TIMESTEP===========================
 100  tsteps= tsteps+1
      e2p=exp(2.d0*a/rp)
c find e2ppr=e2p*(phi,r(rp-)+phi,r(rp+))/2
      e2ppr=-e2p*.5d0*a/rp**2
c ----------particle-----------------------------------------
c evolve with 4-th order Runge-Kutta
      call runge4(am2,e2p,e2ppr,dt,rp,utlr,rp,utlr)
      if (rp.le.0.d0) then
         print *,'particle fallen in to the origin !!!'
         stop
      endif
c ----------field----------------------------------------------
      call phi1(mr,rp,a,up)

c write fort.18 :[t,r(t)] and fort.19 :[t,utt(t),utlr(t)]
      write(18,*) tsteps*dt,rp,a/rp
c calculate utt
      utt=sqrt(e2p+utlr**2+am2/rp**2)
      write(19,*) rp,utt,utlr
      if(tsteps.eq.dimt) goto 200
      goto 100

c estimate the final equilibrium radius erp
 200  call eqrp1(utlp,mr,erp,ea)
c write erp and the field at erp (ea/erp)
      print*,erp,ea/erp

      stop
      end
```