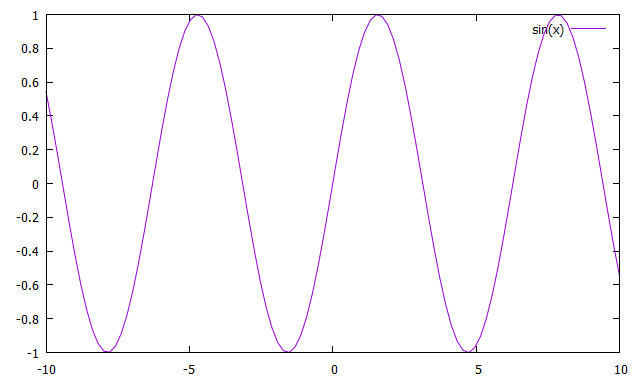
1. The first graph: sin(x) to test gnuplot.



The following graph is the plot file with lines with the data points of:

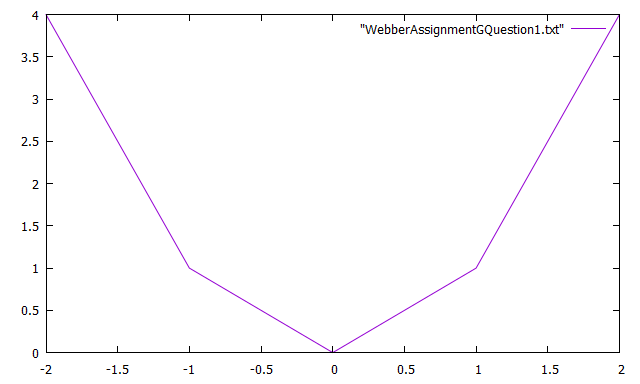
-2 4

-1 1

0 0

1 1

1. 4



1. Use at least three numerical methods to solve:

for . For each method obtain at least 100 data points on your approximation and produce a graph for each method.

Below are the subroutines used by the three numerical methods, they include computef, euler, and extroplate:

subroutine computef(neq, t, y, f)

implicit none

integer :: neq

double precision :: f(neq)

double precision, intent(in) :: t, y(neq)

! This subroutine computes the RHS of the differential equation:

! dy

! -- = f(t, y)

! dt

! where y is the array of unknowns, indexed from one to neq and f is the array of right sides.

!y(0) = 0

f(1) = (t \* dexp(3.0d0 \* t) - 2.0d0 \* y(1))

return

end

subroutine euler(neq, tin, tout, nsteps, yin, yout)

implicit none

integer :: i

integer, intent(in) :: neq, nsteps

double precision :: yout(neq), f(neq), dt

double precision, intent(in) :: tin, tout, yin(neq)

yout = yin

dt = ((tout - tin)/dble(nsteps))

do i = 0, (nsteps - 1)

call computef(neq, (tin + dble(i) \* dt), yout, f)

yout = (yout + dt \* f)

end do

return

end

subroutine extrapolate(neq, tin, tout, yin, yout)

implicit none

integer, intent(in) :: neq

integer :: i, k, nrich, nsteps

double precision :: yout(neq), bot

double precision, intent(in) :: tin, tout, yin(neq)

double precision, allocatable, dimension(:, :, :) :: table

nrich = 10

nsteps = 1

allocate(table(0:nrich, 0:nrich, neq))

do i = 0, nrich

call euler(neq, tin, tout, nsteps, yin, yout)

table(i,0,:) = yout

nsteps = (nsteps \* 2)

end do

do k = 1, nrich

bot = (2.0d0\*\*k - 1.0d0)

do i = k, nrich

table(i,k,:) = (table(i,k-1,:) + (table(i,k-1,:) - table(i-1,k-1,:)) / bot)

end do

end do

yout = table(nrich, nrich, :)

return

end

First method used (euler):

program euler

implicit none

integer :: i, neq, nsteps

double precision :: a, b, dt, tin, tout

double precision, allocatable, dimension(:, :) :: archive

double precision, allocatable, dimension(:) :: t, u, f1, yin, yout

neq = 1

a = 0.0d0

b = 2.0d0

nsteps = 1000 ! at least 100

allocate(yin(neq), yout(neq), t(0:nsteps), u(neq), f1(neq), archive(neq,0:nsteps))

dt = ((b - a) / dble(nsteps))

do i = 0, nsteps

t(i) = (a + dble(i) \* dt)

end do

archive (:,0) = (/ 1.0d0, 0.0d0, 0.0d0, 1.0d0 /) ! initial state vector

yin = archive(:,0)

do i = 0, (nsteps - 1)

tin = t(i)

tout = t(i + 1)

yin = archive(:,i)

call computef(neq, tin, yin, f1)

yout = (yin + (tout - tin) \* f1)

archive(:,i+1) = yout

end do

print\*, 'creating "out5.2.1"'

open(unit=8, file='out5.2.1', status='replace')

do i = 0, nsteps

write(8,\*) t(i), archive(1,i)

end do

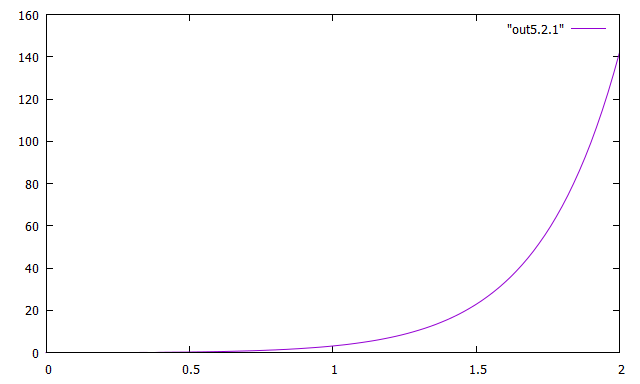
close(8)

deallocate(yin, yout, t, u, f1, archive)

stop

end program euler

Graph of the euler method:



Second method used (extrapolation):

program extrapall

implicit none

integer :: i, neq, nsteps

double precision :: a, b, dt, tin, tout

double precision, allocatable, dimension(:, :) :: archive

double precision, allocatable, dimension(:) :: t, f1, yin, yout

neq = 1

a = 0.0d0

b = 2.0d0

nsteps = 1000

allocate(yin(neq), yout(neq), t(0:nsteps), f1(neq), archive(neq,0:nsteps))

dt = ((b - a) / dble(nsteps))

do i = 0, nsteps

t(i) = (a + dble(i) \* dt)

end do

archive (:,0) = (/ 1.0d0, 0.0d0, 0.0d0, 1.0d0 /) ! initial state vector

yin = archive(:,0)

do i = 0, (nsteps - 1)

tin = t(i)

tout = t(i + 1)

yin = archive(:,i)

call extrapolate(neq, tin, tout, yin, yout)

archive(:,i+1) = yout

end do

print\*, 'Creating "out5.2.2"'

open(unit=8, file='out5.2.2', status='replace')

do i = 0, nsteps

write(8,\*) t(i), archive(1,i)

end do

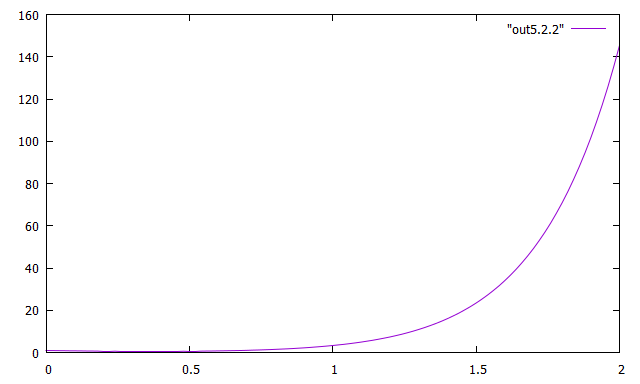
close(8)

deallocate(yin, yout, t, f1, archive)

stop

end program extrapall

Graph of the extrapall method:



Third method used (predictor corrector):

program aopc

implicit none

integer :: i, neq, nsteps

double precision :: a, b, dt

double precision, allocatable, dimension(:, :) :: archive

double precision, allocatable, dimension(:) :: f, t, u, f1, fi, fim1, fim2, fim3, fim4, yin, yout

neq = 1

a = 0.0d0

b = 2.0d0

nsteps = 100

allocate(yin(neq), yout(neq), u(neq), t(0:nsteps), f(neq), archive(neq,0:nsteps))

allocate(fi(neq), fim1(neq), fim2(neq), fim3(neq), fim4(neq))

dt = ((b - a) / dble(nsteps))

do i = 0, nsteps

t(i) = (a + dble(i) \* dt)

end do

archive (:,0) = (/ 1.0d0, 0.0d0, 0.0d0, 1.0d0 /) ! initial state vector

call abinitializer(neq, nsteps, t, archive)

do i = 4, (nsteps - 1)

call computef(neq, t(i), archive(:,i), fi)

call computef(neq, t(i-1), archive(:,i-1), fim1)

call computef(neq, t(i-2), archive(:,i-2), fim2)

call computef(neq, t(i-3), archive(:,i-3), fim3)

call computef(neq, t(i-4), archive(:,i-4), fim4)

u = archive(:,i) + dt / 720.0d0 \* (1901.0d0 \* fi - 2774.0d0 \* fim1 + 2616.0d0 \* fim2 - 1274.0d0 \* fim3 + 251.0d0 \* fim4)

end do

print\*, 'Creating "out5.2.3"'

open(unit=8, file='out5.2.3', status='replace')

do i = 0, nsteps

write(8,\*) t(i), archive(1,i)

end do

close(8)

deallocate(yin, yout, u, f, t, f1, fi, fim1, fim2, fim3, fim4, archive)

stop

end program aopc

Abinitializer function used by aopc:

subroutine abinitializer(neq, nsteps, t, archive)

implicit none

integer :: i, nsteps

integer, intent(in) :: neq

double precision, intent(in) :: t(0:nsteps)

double precision :: tin, tout, yin(neq), yout(neq), archive(neq,0:nsteps)

do i = 0, neq

tin = t(i)

yin = archive(:,i)

tout = t(i+1)

call extrapolate(neq, tin, tout, yin, yout)

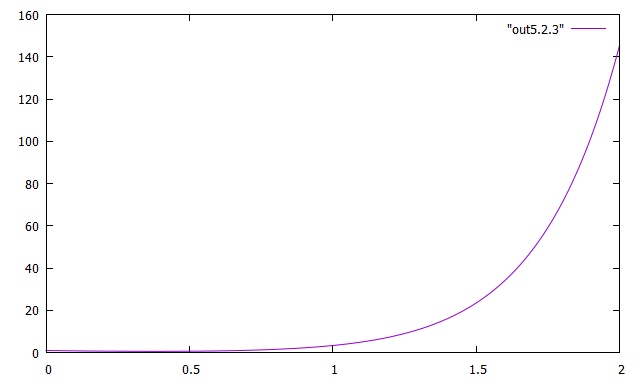
archive(:,i+1) = yout

end do

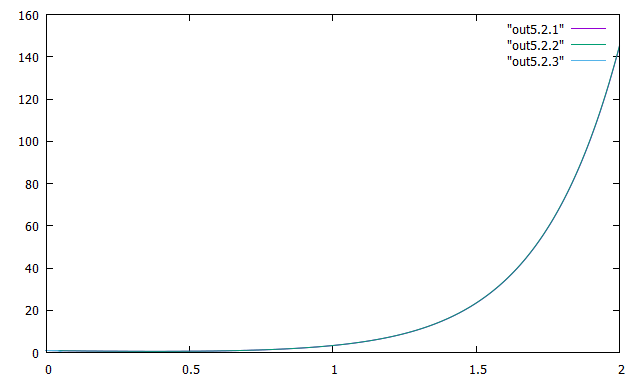
return

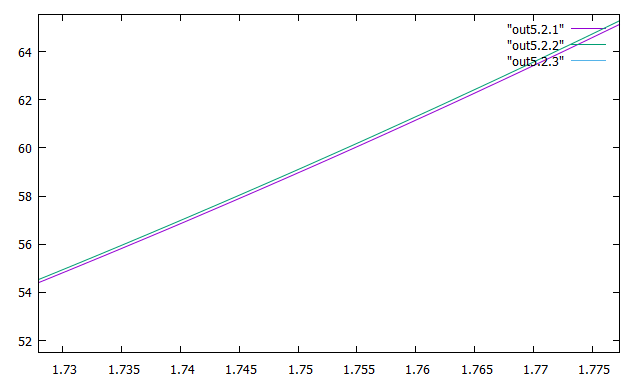
end

Graph of the predictor corrector method:



Graph of methods 1, 2, and 3 along with a zoomed in version:





1. In the orbit problem, we had initial *x* velocity zero and initial *y* velocity one. Experiment with changing the initial *y* velocity to other values. Discuss your results. Reading about Kepler’s laws may be informative. Provide some graphs of interesting orbits.

program WebberHomeworkGQuestion3

implicit none

integer :: neq, nsteps, i

double precision, allocatable, dimension(:) :: yin, yout, t, f1, f2, u

double precision, allocatable, dimension(:, :) :: archive

double precision :: tin, tout, dt, a, b, x, y, xactual, yactual

neq = 4

nsteps = 10000

allocate(yin(neq), yout(neq), t(0:nsteps), u(neq), f1(neq), f2(neq), archive(neq, 0:nsteps))

a = 0.0d0

b = 3.0d1

dt = (b - a) / dble(nsteps)

do i = 0, nsteps

t(i) = a + dble(i) \* dt

end do

archive (:,0) = (/ 1.0d0, 0.0d0, 0.0d0, 1.0d0 /)

yin = archive(:,0)

do i = 0, (nsteps - 1)

tin = t(i)

tout = t(i + 1)

yin = archive(:,i)

call computef(neq, tin, yin, f1)

u = (yin + (tout - tin) \* f1)

call computef(neq, tout, u, f2)

yout = yin + (tout - tin) \* (f1 + f2) / 2.0d0 ! euler

archive(:,i+1) = yout

end do

print\*,'Creating "out0" '

open(unit=8,file='q3out', status='replace')

do i = 0, nsteps

write(8,\*) archive(1,i), archive(2,i)

end do

close(8)

print\*,'writing eulererror'

open(unit=8, file='eulererror', status='replace')

do i = 0, nsteps

x = archive(1,i)

y = archive(2,i)

xactual = cos(t(i))

yactual = sin(t(i))

write(8,\*) i, sqrt((x - xactual) \*\* 2 + (y - yactual) \*\* 2)

end do

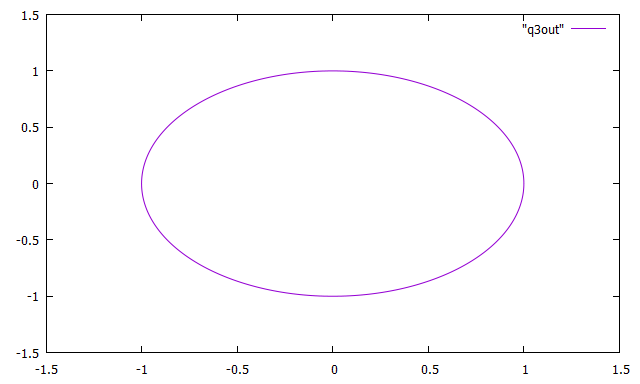
close(8)

deallocate(yin, yout, archive, t, f1, f2)

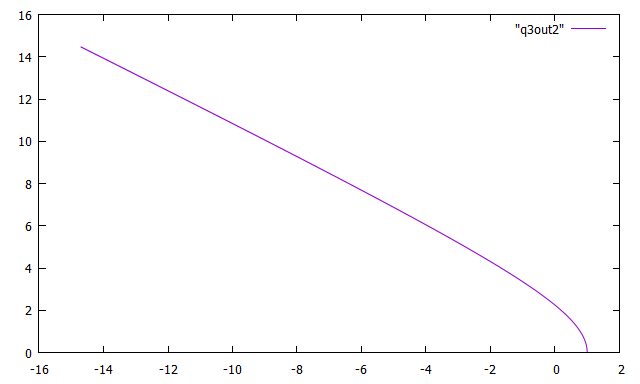
stop

end program WebberHomeworkGQuestion3

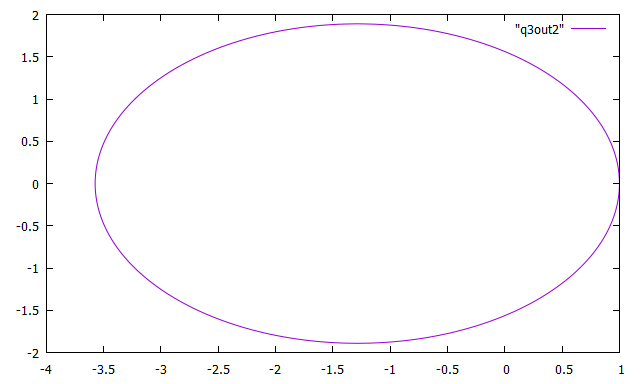
Graph for the initial state (/ 1.0d0, 0.0d0, 0.0d0, 1.0d0 /):



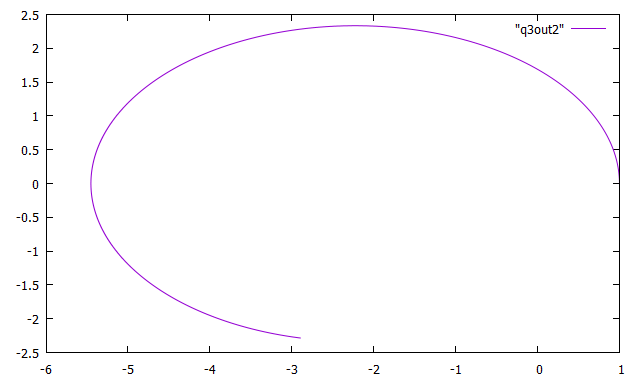
Graph for the initial state (/ 1.0d0, 0.0d0, 0.0d0, 1.5d0 /):



Graph for the initial state (/ 1.0d0, 0.0d0, 0.0d0, 1.25d0 /):



Graph for the initial state (/ 1.0d0, 0.0d0, 0.0d0, 1.3d0 /):



After changing the initial state, I could see that the orbit was becoming less stable and the variable tended towards 1.3d0 as pictured above. Once it reached this value it was no longer a circular shape or stable orbit.