Inf2320 Chapter 6

17th November 2003

Exercises

6.1+6.2

Algorithm for solving quadratic equations, with input data checks.

Quadratic (a, b, c)
  if $a = 0$ and $b = 0$ and $c \neq 0$
    write error message
  else if $a = 0$
    return $-c/b$
  else
    $r_1 = (-b + \text{sqrt}(b^2 - 4ac))/(2a)$
    $r_2 = (-b - \text{sqrt}(b^2 - 4ac))/(2a)$
    if $r_1 = r_2$
      return r_1
    else
      return r_1, r_2
    end if
  end if

6.3

Algorithm for solving quadratic equations, not using imaginary numbers.

Quadratic (a, b, c)
  if $a = 0$ and $b = 0$ and $c \neq 0$
    write error message
  else if $a = 0$
    return $-c/b, 0$
  else if $b^2 \geq 4ac$
    $r_1 = (-b + \text{sqrt}(b^2 - 4ac))/(2a), 0$
    $r_2 = (-b - \text{sqrt}(b^2 - 4ac))/(2a), 0$
    if $r_1 = r_2$
      return r_1
    else
      return r_1, r_2
  else
    $r_1 = -b/(2a), \text{sqrt}(4ac - b^2)$
    $r_2 = -b/(2a), -\text{sqrt}(4ac - b^2)$
    if $r_1 = r_2$
      return r_1
    else
      return r_1, r_2
  end if

6.5

Use the following pseudocode:

$\Delta t = T/N$
$u_0 = U_0$
write $(0, u_0)$ to file
for $i = 1, \ldots, N$
  $u_i = \text{heun}(f, u_{i-1}, \Delta t, 1)$
  write $(i\Delta t, u_i)$ to file
end for

The number of f-function evaluations is now $3N$ instead of $3 \sum_{i=1}^{N} i$.

6.6

Generalized Heun's method.

heun($f, U_0, \Delta t, N$)
  $u = U_0$
  for $n = 0, \ldots, N-1$
    $u \leftarrow u + \frac{\Delta t}{2}\left[f(u, n\Delta t) + f(u + \Delta t f(u, n\Delta t), (n+1)\Delta t)\right]$
  end for
  return u

Projects

6.7.1; Computing the volume of a Cylindrical Container

a)

$$\begin{aligned} V &= \int_0^H \int_0^{2\pi} \int_0^{R(x)} r\, dr\, d\theta\, dx \\ &= \int_0^H \int_0^{2\pi} \frac{1}{2}\left[R(x)\right]^2 d\theta\, dx \\ &= \int_0^H \left[\frac{\theta}{2}\left[R(x)\right]^2\right]_{\theta=0}^{2\pi} dx \\ &= \pi \int_0^H \left[R(x)\right]^2 dx \end{aligned}$$

b)

R(x)
  if $x \leq 3$ then $R = x$
  else if $3 \leq x \leq 5$ then $R = 3$
  else if $5 \leq x \leq 6$ then $R = 13 - 2x$
  else if $6 \leq x$ then $R = 1$
  Return R

c)

volume(R(x),H,n)
  $x = 0$
  V=$\frac{1}{2}\left(\left[R(0)\right]^2 + \left[R(H)\right]^2\right)$
  for $i = 1, \ldots, n-1$
    $x = x + Dx$
    $V = V + \left[R(x)\right]^2$
  end for
  V=V*3.14
  Return V

d) Se the program **p61.c**.

e) Se the program **p61e.c**. This program uses Algorihm 6.2. from the book. This version is easier to modify for other applications.

6.7.2; A Class Hierarchy for Scalar ODE Solvers

a)
```
class ODESolver_prm
    data:
      u0, f, Dt
      tstop
      method
      ...
    methods
      constructor()
      create()
      default()
      read()
      write()
```

b)
```
class ODESolver
    data:
      prm
    methods:
      constructor(ODESolver_prm p)
        prm = p
      calculate()
```

c)
```
class Heun, subclass of ODESolver
    data:
    methods:
      constructor(ODESolver_prm p)
        ODESolver.constructor.(p)
        calculate()
          u[0]=u0
          for i=1,..,n
            v=f(u[i])
            u[i+1]=u[i]+dt(v*f(u[i] + dt*v ))/2
          end for
          return u
```

d)-f) For implementatons in Python see **ODE.py**. The Java and C++ classes are shown in their own sections. (The C++ classes use a slightly different scheme.)

g) We can test the programs using the problem from exercise 2.1; $u'(t) = 1+4r(t),\ r(0) = 0$. We know that the analythical solution is

$$r(t) = \frac{1}{4}(e^{4t} - 1)$$

which gives the results $r(0.5) = 1.597$ and $r(1) = 13.400$. We compare this to the results we get by running **p62.py**:

```
>python p62.py -d 0.01 -s 1 -u 0  -f '1+4*u' -m RK
Dt=0.01 tstop=1 npoints=100 u0=0 f='1+4*u'  method=RK
13.3995310153

>python p62.py -d 0.01 -s 0.5 -u 0  -f '1+4*u' -m Heun
Dt=0.01 tstop=0.5 npoints=50 u0=0 f='1+4*u'  method=Heun
1.596308904
```

6.7.3; Software for Systems of ODEs

a)

Heun's method for systems of ODEs,
storing computed values in an array.

heun$(f, g, Y_0, Z_0, \Delta t, N)$
  $y_0 = Y_0$
  $z_0 = Z_0$
  for $n = 0, \ldots, N-1$
    $v = f(y_n, z_n)$
    $w = g(y_n, z_n)$
    $y_{n+1} = y_n + \frac{\Delta t}{2}\left[v + f(y_n + \Delta t v, z_n + \Delta t w)\right]$
    $z_{n+1} = z_n + \frac{\Delta t}{2}\left[w + g(y_n + \Delta t v, z_n + \Delta t w)\right]$
  end for
  return $(y_0, y_1, \ldots, y_N), (z_0, z_1, \ldots, z_N)$

Heun's method for systems of ODEs,
returning only the final values.

heun$(f, g, Y_0, Z_0, \Delta t, N)$
  $y_0 = Y_0$
  $z_0 = Z_0$
  for $n = 0, \ldots, N-1$
    $v = f(y, z)$
    $w = g(y, z)$
    $y2 = y + \frac{\Delta t}{2}\left[v + f(y + \Delta t v, z + \Delta t w)\right]$
    $z2 = z + \frac{\Delta t}{2}\left[w + g(y + \Delta t v, z + \Delta t w)\right]$
    $y = y2$
    $z = z2$
  end for
  return (y, z)

b)+c) For implementation in Python, see **ODEs.py**.

d) We can test this program using the problem from exercise **3.1**. Simply run **p63.py** a few times with different values for *tstop*, e.g..

```
>python p63.py -d 0.001 -s 2 -y 1.9 -z 0.1  -f '(2-v)*u' -g '(u-1)*v'
```

and compare the results with the arrays generated by **ex31.m**.

Programs (C,C++, Python, Java)

p61.c

```
/*  A simple c program showing implementations of the
 *  functions from project 6.1.
 */
#include <stdio.h>

double R(double x);
double volume(double H, int n, double (*Rfunk)(double x));

int main (int argc, char *argv[]) {
  double result;

  result = volume(8, 1000, R);
  printf("Volume: %lf\n", result);
}

double volume(double H, int n, double (*Rfunk)(double x)){
  double Dx, V, x;
  int i;
  Dx=H/n;
  x=0;
   printf("Dx: %lf\n", Dx);
   V=0.5*((*Rfunk)(0)*(*Rfunk)(0)+(*Rfunk)(H)*(*Rfunk)(H));
   for (i=1; i<n; i++){
     x=x+Dx;
     V=V+(*Rfunk)(x)*(*Rfunk)(x);
     }
    V=3.14159*Dx*V;
    return V;
}

double R(double x){
  double R;
  if (x <= 3) R=x;
  else if  (x <= 5) R=3;
  else if  (x <= 6)  R=13-2*x;
  else  R=1;
  return R;
}
```

p61e.c

```
/*  This program does the same as program p61.c,
 *  but using Algorithm 6.2.
 */
#include <stdio.h>
```

```
double f(double x);
double trapezoidal(double a, double b, int n, double (*function)(double x));
double R(double x);

int main (int argc, char *argv[]) {
  double result;

  result = trapezoidal(0, 8, 100, f);
  printf("Volume: %lf\n", result);
}

double trapezoidal(double a, double b, int n, double (*function)(double x)){
   double h, s, x;
   int i;
   h=(b-a)/n;
   s=0;
   x=a;
   for (i=1; i<n; i++){
     x=x+h;
     s=s+(*function)(x);
       }
   s=s+0.5*((*function)(a)+(*function)(b));
   s=h*s;
   return s;
}

double R(double x){
  double R;
  if (x <= 3) R=x;
  else if  (x <= 5) R=3;
  else if  (x <= 6)  R=13-2*x;
  else  R=1;
  return R;
}

double f(double x){
  double res;
  res= 3.1415*R(x)*R(x);
  return res;
}
```

ODE.py

```
import sys
from Numeric import *


class ODE:
    """
    Base class for numerical methods for ODEs.
```

```
    """

    def __init__(self, p):
        self.prm = p  # ODE_prm object

    def calculate(self):
        """Perform integration."""
        # to be implemented in subclasses
        return "ODE.calculate; not impl. in subclass"

class Euler(ODE):
    """
    Euler's method.
    """
    def __init__(self, p):
        ODE.__init__(self, p)

    def calculate(self):
        p = self.prm; f = p.f; n = p.npoints; u0 = p.u0; Dt=p.dt;
        u = zeros(n+1, Float)
        u[0]=u0  #initial condition
        # advance n steps:
        for i in range (0,n,1):
            u[i+1] = u[i]+Dt*f(u[i])
        return u

class RK(ODE):
    """
    4. Order Runge-Kutta.
    """
    def __init__(self, p):
        ODE.__init__(self, p)

    def calculate(self):
        p = self.prm; f = p.f; n = p.npoints; u0 = p.u0; Dt=p.dt;
        u = zeros(n+1, Float)
        u[0]=u0  #initial condition
        # advance n steps:
        for i in range (0,n,1):
            F1 = f(u[i])
            F2 = f(u[i]+Dt*0.5*F1)
            F3 = f(u[i]+Dt*0.5*F2)
            F4 = f(u[i]+Dt*F3)
            u[i+1]=u[i] + Dt*(F1 + 2*F2 + 2*F3 +F4)/6
        return u

class Heun(ODE):
    """
    Heun's method, 2. order Runge-Kutta.
    """
```

```
    def __init__(self, p):
        ODE.__init__(self, p)

    def calculate(self):
        p = self.prm; f = p.f; n = p.npoints; u0 = p.u0; Dt=p.dt;
        u = zeros(n+1, Float)
        u[0]=u0  #initial condition
        # advance n steps:
        for i in range (0,n,1):
            v = f(u[i])
            u[i+1] = u[i]+0.5*Dt*(v+f(u[i] + Dt*v))
        return u


class ODE_prm:
    """
    Holds all parameters needet to initiate objects
    in the ODE hierarcy.
    """

    def __init__(self):
        self.default()
        return

    def default(self):
        """assign appropriate default values to all parameters"""
        self.tstop = 1
        self.u0 = 1
        self.dt = 0.01
        self.f = Function('1+x', independent_variable='x')
        self.method = "Heun"
        self.npoints = int(self.tstop/self.dt)
    def read(self, argv=sys.argv[1:]):
        # manual parsing
        for i in range(0,len(argv),2):
            if argv[i] == "-s":  self.tstop = float(argv[i+1])
            if argv[i] == "-d":  self.dt = float(argv[i+1])
            if argv[i] == "-f":  self.f = Function(argv[i+1])
            if argv[i] == "-m":  self.method = argv[i+1]
            if argv[i] == "-u":  self.u0 = float(argv[i+1])
        self.npoints = int(self.tstop/self.dt)
    def write(self):
        print "Dt=%g tstop=%g npoints=%g u0=%g f='%s'  method=%s" % \
              (self.dt, self.tstop, self.npoints, self.u0, \
               self.f.__name__, self.method)

    def create(self):
        """Create subclass of ODE"""
        code = "i = %s(self)" % self.method
        exec(code)  # turn string into Python code
```

```
        return i

class Function:
    """
    Unified treatment of functions; strings or function objects.
    Examples on usage:

        def myfunc(u):
            return 1+x

        f = Function(myfunc, 'x')  # attach Python function
        v = f(1.2)

        f = Function('1+t', 't')   # specify function by string
        v = f(1.2)
    """
    def __init__(self, f, independent_variable='u'):
        self.f = f  # expression or function object
        self.var = independent_variable  # 'x', 't' etc.
        if type(f) == type(""):
            self.f_is_string = 1
            self.__name__ = self.f
        else:
            # function object:
            self.f_is_string = 0
            self.__name__ = self.f.__name__
    def __call__(self, u):
        if self.f_is_string:
            exec("%s = %g" % (self.var, u))
            return eval(self.f)
        else:
            return self.f(u)
```

p62.py

```
#!/usr/bin/env python
from ODE import *   # give access to library
from math import *
from sys import *

o=ODE_prm()
o.read(sys.argv[1:])
o.write()
i=o.create()
result = i.calculate()
print result[-1]
```

Java classes for Project 6.7.2

```
class ODESolver {
```

```
    public ODESolver_prm prm;

    public ODESolver (ODESolver_prm p)
    { prm = p; }

    public double [] calculate ()
    {
System.out.println("ODE.calculate; not impl. in subclass");
        double k [] = new double[1];
return k;
    }
}

class Heun extends ODESolver {
    public Heun (ODESolver_prm p)
    { super(p); }

    public double [] calculate ()
    {
        double dt = prm.dt; double u0 = prm.u0; int n = prm.n;
Func f = prm.f;
double u [] = new double[n+1];
        u[0] = u0; // initial condition

        // advance n steps:
        for (int i = 0; i < n; i++) {
          double v=f.f(u[i]);
          u[i+1] = u[i] + 0.5*dt*(v + f.f( u[i] + dt*v ));
}
return u;
    }
}
class Euler extends ODESolver {
    public Euler (ODESolver_prm p)
    { super(p); }

    public double [] calculate ()
    {
        double dt = prm.dt; double u0 = prm.u0; int n = prm.n;
Func f = prm.f;
double u [] = new double[n+1];
        u[0] = u0; // initial condition

        // advance n steps:
        for (int i = 0; i < n; i++) {
          u[i+1] = u[i] + dt*f.f(u[i]);
}
return u;
    }
}
```

```
class RK extends ODESolver {
    public RK (ODESolver_prm p)
    { super(p); }

    public double [] calculate ()
    {
        double dt = prm.dt; double u0 = prm.u0; int n = prm.n;
Func f = prm.f;
double u [] = new double[n+1];
        u[0] = u0; // initial condition

        // advance n steps:
        for (int i = 0; i < n; i++) {
          double F1 = f.f(u[i]);
          double F2 = f.f(u[i] + dt*0.5*F1);
          double F3 = f.f(u[i] + dt*0.5*F2);
          double F4 = f.f(u[i] + dt*F3);
          u[i+1] = u[i] + dt*(F1 + 2*F2 + 2*F2 + F4)/6;
}
return u;
    }
}
class ODESolver_prm {
    public double u0, dt, tstop;
    public Func f;
    public int n;
    public String method;

    public ODESolver_prm ()
    {   /* default values */
u0=1; dt=0.01; tstop=1;  f = new f2();
method = "Heun";
        n=  (int) (tstop/dt);
    }

    public void read (String argv[])
    {
int i;
for (i = 0; i < argv.length; i=i+2) {
    if (argv[i].compareTo("-s") == 0) {
tstop = Double.valueOf(argv[i+1]).doubleValue();
    }
    if (argv[i].compareTo("-d") == 0) {
dt = Double.valueOf(argv[i+1]).doubleValue();
    }
    if (argv[i].compareTo("-u") == 0) {
        u0 = Integer.valueOf(argv[i+1]).intValue();
    }
    if (argv[i].compareTo("-m") == 0) {
method = argv[i+1];
```

```
    }
}
        n=  (int) (tstop/dt);
    }

    public void write ()
    {  System.out.println("Dt=" + dt + " tstop=" + tstop +
                          " u0=" + u0  + " n=" + n +
  " method=" + method);
    }

    public ODESolver create ()
    {
ODESolver i;
if (method.compareTo("Heun") == 0) {
    i = new Heun(this);
}
else if (method.compareTo("Euler") == 0) {
            i = new Euler(this);
        }
else if (method.compareTo("RK") == 0) {
            i = new RK(this);
        } else {
    i = new Heun(this);
}
return i;
    }
}
```

C++ classes for Project 6.7.2

```
class ODESolver_prm
{
public:
  char         method[30];  // name of subclass in ODESolver hierarchy
  ODEProblem*  problem;     // pointer to user's problem class
  ODESolver*   create ();   // create correct subclass of ODESolver
};

ODESolver* ODESolver_prm::create ()
{
  ODESolver* ptr = NULL;
  if      (strcmp(method, "ForwardEuler") == 0)
    ptr = new ForwardEuler (problem);
  else if (strcmp(method, "RungeKutta4") == 0)
    ptr = new RungeKutta4 (problem);
  else if (strcmp(method, "Heun4") == 0)
    ptr = new Heun4 (problem);
  else {
```

```
    cout << "\n\nODESolver_prm::create:\n\t"
         <<  "Method " << method << " is not available\n\\n";
    exit(1);
  }
  return ptr;
}
class ODESolver
{
protected:             // members only visible in subclasses
  ODEProblem* eqdef;   // definition of the ODE in user's class
public:                // members visible also outside the class
  ODESolver (ODEProblem* eqdef_)
    { eqdef = eqdef_; }
  virtual ~ODESolver () {} // always needed, does nothing here...
  virtual void init() {}   // initialize solver data structures
  virtual void advance (MyArray<double>& y, double& t, double& dt);
};
class ODESolver
{
protected:             // members only visible in subclasses
  ODEProblem* eqdef;   // definition of the ODE in user's class
public:                // members visible also outside the class
  ODESolver (ODEProblem* eqdef_)
    { eqdef = eqdef_; }
  virtual ~ODESolver () {} // always needed, does nothing here...
  virtual void init() {}   // initialize solver data structures
  virtual void advance (MyArray<double>& y, double& t, double& dt);
};

#endif


void ODESolver::advance(MyArray<double>& /*y*/,
                        double& /*t*/, double& /*dt*/)
{
  cout << "\n\nODESolver::advance:\n\t"
       << "This function must be implemented in a subclass!\n\n";
}
class ForwardEuler : public ODESolver
{
  MyArray<double> scratch1;    // needed in the algorithm
public:
  ForwardEuler (ODEProblem* eqdef_);
  virtual void init ();  // for allocating scratch1
  virtual void advance (MyArray<double>& y, double& t, double& dt);
};
ForwardEuler::ForwardEuler(ODEProblem* eqdef_) : ODESolver(eqdef_) {}

void ForwardEuler:: init ()
{
```

```
  // allocate internal data structure:
  scratch1.redim (eqdef->size());
}

void ForwardEuler::advance(MyArray<double>& y, double& t, double& dt)
{
  eqdef->equation (scratch1, y, t);  // evaluate scratch1 (as f)
  const int n = y.size();
  for (int i = 1; i <= n; i++)
    y(i) += dt * scratch1(i);
  t += dt;
}
class RungeKutta4 : public ODESolver
{
  MyArray<double> scratch1, scratch2, scratch3; // needed in algorithm
public:
  RungeKutta4 (ODEProblem* eqdef_);
  virtual void init ();
  virtual void advance (MyArray<double>& y, double& t, double& dt);
};
RungeKutta4::RungeKutta4(ODEProblem* eqdef_) : ODESolver (eqdef_) {}

void RungeKutta4:: init ()
{
  // allocate internal data structure:
  const int n = eqdef->size();
  scratch1.redim (n); scratch2.redim (n); scratch3.redim (n);
}

void RungeKutta4:: advance (MyArray<double>& y, double& t, double& dt)
{
  const double dt2 = 0.5*dt;
  const double dt6 = dt/6.0;
  const int n = y.size();
  eqdef->equation (scratch1, y, t);
  int i;
  for (i = 1; i <= n; i++)
    scratch2(i) = y(i)  +  dt2 * scratch1(i);
  eqdef->equation (scratch1, scratch2, t+dt2);
  for (i = 1; i <= n; i++)
    scratch2(i) = y(i)  +  dt2 * scratch1(i);
  eqdef->equation (scratch3, scratch2, t+dt2);
  for (i = 1; i <= n; i++)
    {
      scratch2(i) = y(i)   +  dt * scratch3(i);
      scratch3(i) = scratch1(i) + scratch3(i);
    }
  eqdef->equation (scratch1, scratch2, t+dt);
  eqdef->equation (scratch2, y, t);
```

```
  for (i = 1; i <= n; i++)
    y(i) = y(i) + dt6*(scratch1(i) + scratch2(i) + 2*scratch3(i));
  t += dt;
}

class Heun : public ODESolver
{
  MyArray<double> scratch1, scratch2, scratch3; // needed in algorithm
public:
  Heun (ODEProblem* eqdef_);
  virtual void init ();
  virtual void advance (MyArray<double>& y, double& t, double& dt);
};
Heun::Heun(ODEProblem* eqdef_) : ODESolver (eqdef_) {}

void Heun:: init ()
{
  // allocate internal data structure:
  const int n = eqdef->size();
  scratch1.redim (n); scratch2.redim (n); scratch3.redim (n);
}

void Heun:: advance (MyArray<double>& y, double& t, double& dt)
{
  const double dt2 = 0.5*dt;
  const int n = y.size();
  eqdef->equation (scratch1, y, t);
  int i;
  for (i = 1; i <= n; i++)
    scratch2(i) = y(i)  + dt *scratch1(i);
  eqdef->equation (scratch3, scratch2, t);
  for (i = 1; i <= n; i++)
    y(i) = y(i) + dt2*(scratch1(i) + scratch3(i));
  t += dt;
}
```

The MyArray template class, used in the C++ classes.

```
template <class T>
class MyArray
{
protected:
  T*      A;                              // vector entries (C-array)
  int     length;
  void    allocate (int n);               // allocate memory, length=n
  void    deallocate();                   // free memory
public:
  MyArray ();                             // MyArray<T> v;
  MyArray (int n);                        // MyArray<T> v(n);
```

```
  MyArray (const MyArray<T>& w);          // MyArray<T> v(w);
 ~MyArray ();                             // clean up dynamic memory

  int redim (int n);                      // v.redim(m);
  int size () const { return length; }    // n = v.size();

  void operator= (const MyArray<T>& w);   // v = w;
  T  operator() (int i) const;            // a = v(i);
  T& operator() (int i);                  // v(i) = a;

  void print (ostream& o) const;          // v.print(cout);
};
template<class T>
void MyArray<T>::allocate(int n)
{
  length = n;
  A = new T[n];
}

template<class T>
void MyArray<T>::deallocate()
{
  delete [] A;
  length = 0;
}

template<class T> MyArray<T>::MyArray()
{ A = NULL; length=0; }

template<class T> MyArray<T>::MyArray(int n)
{ allocate(n); }

template<class T> MyArray<T>::MyArray(const MyArray<T>& w)
{
  allocate(w.size());
  *this = w;
};

template<class T> MyArray<T>::~MyArray()
{ deallocate(); }

template<class T>
int MyArray<T>::redim(int n)
{
  if (length==n)
    return 0;
  if (A!= NULL)
    deallocate();
  allocate(n);
  return 1;
```

```
};

template<class T>
void MyArray<T>::operator=(const MyArray<T> & w)
{
  redim (w.size());
  for (int i=0; i<length; i++)
    A[i] = w.A[i];
}

template<class T>
inline T MyArray<T>::operator()(int i) const
{
#ifdef SAFETY_CHECKS
  if (i < 1 || i > length)
    cout << "MyArray::operator(), illegal index, i=" << i;
#endif
  return A[i-1];
}

template<class T>
inline T& MyArray<T>::operator()(int i)
{
#ifdef SAFETY_CHECKS
  if (i < 1 || i > length)
    cout << "MyArray::operator(), illegal index, i=" << i;
#endif
  return A[i-1];
}

template<class T>
inline ostream& operator<< (ostream& o, const MyArray<T>& v)
{ v.print(o); return o; }

template<class T>
void MyArray<T>::print(ostream& o) const
{
  for (int i=1; i<=length; i++)
    o << (*this)(i) << ((i==length) ? "" : " ");
}
```

A sample problem, for testing the C++ program.

```
class Simprob : public ODEProblem
{
protected:
  double c1,c2,c3,c4,omega;  // problem dependent paramters
public:
  Simprob () {}
```

```
  virtual void equation (MyArray<double>& f,
                         const MyArray<double>& y, double t);
  virtual int  size () { return 1; }  // 2x2 system of ODEs
  virtual void scan ();
  virtual void print (ostream& os);
};


void Simprob::equation(MyArray<double>& f,
                         const MyArray<double>& y, double t)
{
  f(1) = c1+c2*y(1);
}

void Simprob:: scan ()
{
  // first we need to do everything that ODEProblem::scan does:
  ODEProblem::scan();
  // additional reading here:
  cout << "Give c1, c2: ";
  cin >> c1 >> c2;
  print(cout);  // convenient check for the user
}

void Simprob:: print (ostream& os)
{
  // print everything that ODEProblem::print prints:
  ODEProblem::print(os);
  // print the equation:
  os << "The equation solved by class Simprob reads\n"
     << "dy/dt = " << c1 << " + "<< c2<<"*y\n";
}

int main (int argc, const char* argv[])
{
  Simprob problem;
  problem.scan();      // read input data and initialize
  problem.timeLoop();  // solve problem
}
```

ODEs.py

```
import sys
from Numeric import *


class ODEs:
    """
    Base class for numerical methods for ODEs.
```

```
    """

    def __init__(self, p):
        self.prm = p  # ODEs_prm object

    def calculate(self):
        """Perform integration."""
        # to be implemented in subclasses
        return "ODEs.calculate; not impl. in subclass"


class Heun(ODEs):
    """
    Heun's method, 2. order Runge-Kutta.
    """
    def __init__(self, p):
        ODEs.__init__(self, p)

    def calculate(self):
        p = self.prm; f = p.f; g = p.g; n = p.npoints; y0 = p.y0; z0 = p.z0; Dt
        u = zeros(n+1, Float)
        y=y0  #initial conditions
        z=z0
        # advance n steps:
        for i in range (0,n,1):
            v = f(y,z)
            w = g(y,z)
            y2 = y +Dt*0.5*(v + f(y+Dt*v,z+Dt*w))
            z2 = z +Dt*0.5*(w + g(y+Dt*v,z+Dt*w))
            y=y2
            z=z2
        return (y,z)


class ODEs_prm:
    """
    Holds all parameters needet to initiate objects
    in the ODEs hierarcy.
    """

    def __init__(self):
        self.default()
        return

    def default(self):
        """assign appropriate default values to all parameters"""
        self.tstop = 1
        self.y0 = 1
        self.z0 = 1
        self.dt = 0.01
```

```
        self.f = Function('y+z', independent_variable='y',\
                          independent_variable2='z')
        self.g = Function('y-z', independent_variable='y',\
                          independent_variable2='z')
        self.method = "Heun"
        self.npoints = int(self.tstop/self.dt)
    def read(self, argv=sys.argv[1:]):
        # manual parsing
        for i in range(0,len(argv),2):
            if argv[i] == "-s":  self.tstop = float(argv[i+1])
            if argv[i] == "-d":  self.dt = float(argv[i+1])
            if argv[i] == "-f":  self.f = Function(argv[i+1])
            if argv[i] == "-g":  self.g = Function(argv[i+1])
            if argv[i] == "-m":  self.method = argv[i+1]
            if argv[i] == "-y":  self.y0 = float(argv[i+1])
            if argv[i] == "-z":  self.z0 = float(argv[i+1])
        self.npoints = int(self.tstop/self.dt)
    def write(self):
        print "Dt=%g tstop=%g npoints=%g y0=%g z0=%g f='%s' g='%s' \
               method=%s" % \
              (self.dt, self.tstop, self.npoints, self.y0, self.z0, \
               self.f.__name__, self.g.__name__, self.method)

    def create(self):
        """Create subclass of ODE"""
        code = "i = %s(self)" % self.method
        exec(code)  # turn string into Python code
        return i

class Function:
    """
    Unified treatment of functions; strings or function objects.
    Examples on usage:
      def myfunc(x,y):
         return x+y

      f = Function(myfunc, 'x','y')  # attach Python function
      v = f(2,1)
      print v

      f = Function('1+x-2*y', 'x', 'y')   # specify function by string
      v = f(1,2)
      print v
    """

    def __init__(self, f, independent_variable='u',\
                 independent_variable2='v'):
        self.f = f  # expression or function object
        self.var = independent_variable  # 'x', 't' etc.
        self.var2 =independent_variable2  # 'x', 't' etc.
```

```
        if type(f) == type(""):
            self.f_is_string = 1
            self.__name__ = self.f
        else:
            # function object:
            self.f_is_string = 0
            self.__name__ = self.f.__name__
    def __call__(self, u, h):
        if self.f_is_string:
            exec("%s = %g" % (self.var, u))
            exec("%s = %g" % (self.var2, h))
            return eval(self.f)
        else:
            return self.f(u,h)
```

p63.py

```
#!/usr/bin/env python
from ODEs import *   # give access to library
from math import *
from sys import *

o=ODEs_prm()
o.read(sys.argv[1:])
o.write()
i=o.create()
result = i.calculate()
print result
```