Math321 Numerical Analysis

Tutorial 2, Newton Bairstow for factoring polynomials.

I will give you a copy of the Maple program bairstow.txt which implements Bairstow's algorithm for finding a real quadratic factor $z^2 + p\ z + q$ of a polynomial $P_n(z)$ with real coeficients, as discussed in lectures. The program is incomplete, in that I have left a couple of lines implementing the partial differentiation with respect to q for you to fill in.

The program will be on my floppy, so copy it to the D: drive, and thence to your own floppy. After making corrections as required with Notepad, and saving you can start Maple and execute the program with the statements:

> restart;

> read "A:/bairstow.txt";

As before, most of the statements in my program are terminated with **:** so if you want to see the effect of those particular statements, you should change the ending to a **;** . However, whether you actually see output in response to that change depends on whether the variable *printlevel* is set appropriately, so I tend to use *printf()* statements myself if I want to monitor the progress of a calculation, and save the **;** ending for statements defining the problem.

1. Run the program as entered, with your additions to evaluate the partial derivatives with respect to q, and make sure that the numerical procedure converges to one of the quadratics that Maple itself identifies as one of the factors. If you have made the additions correctly, on this problem you should be able to observe the quadratic convergence of Bairstow's Newton procedure. Verify that the procedure will converge to either of the possible answers, by changing your choice of starting guesses.

2. Change the $pA[]$ list of coefficients to represent the polynomial $(z-1)(z-2)(z-3)(z-4)$. In this situation Bairstow"s method could converge to any one of the 6 different polynomials $(z-1)(z-2) = z^2 - 3z + 2$, $(z-1)(z-3) = z^2 - 4z + 3$, etc. Move the starting guesses around and see what happens. Would you expect anything special to happen if you started with the guess $z^2 - 5z + 5$? Try it and see what actually happens.

3. Change the $pA[]$ list to represent the polynomial $z^4+2z^3+3z^2+2z+1$. Choose whatever starting guess you like, and run the program. Can you explain its behaviour on this problem?

4. Change the definition of np and the coefficient list $pA[]$ to represent the polynomial $z^5 - 1$ and rerun. Notice the factors that Maple itself produces for this problem. Change the guesses and see if you can get the procedure to converge to both of the polynomials. Why are there only two possible polynomials for Bairstow's method to converge to? (You should know from first year the roots of the polynomial $z^5 - 1 = 0$.)

Please ask during the lab if you are having difficulties.

```
# Bairstow's algorithm for finding the roots of polynomials

with (LinearAlgebra): #this package will solve the matrix equation

np := 4;            #the degree of the monic polynomial
pA := array(0..np, [-1, -1, 0, 1, 1]);
#coefficients of increasing powers, starting with z^0
#usually a[np] will be 1, (a monic polynomial)

pN := pA[0]+x*(pA[1]+x*(pA[2]+x*(pA[3]+x*pA[4])));          # the analytic form
fac := factor(pN);                                      # Maple's factors

pstart:= 0.5;            #starting guess for p and q
qstart:= 2.0;

htolerance := 1.0e-6;     #relative accuracy of numerical answer
iterations := 20;         #maximum number, in case someting goes wrong

Bairstow := proc(n, a, pstart, qstart)
    local i, j, p, q, h, dbdp, dbdq, r, drdp, drdq, s,
          dsdp, dsdq, finish, A, b, C, det, hsize, psize;
    global htolerance, iterations;

    j := 1:        #Iteration counter
    p := pstart:
    q := qstart:
    finish := 0;
    while (finish < 1) do
        b[n-1] := 0.0:
        b[n-2] := 1.0:
        dbdp[n-1] := 0.0:
        dbdp[n-2] := 0.0:
        dbdq[n-1] := 0.0:
        dbdq[n-2] := 0.0:

        for i from n-3 by -1  to 0 do
            b[i] := evalf(a[i+2] - p*b[i+1] - q*b[i+2]):
            dbdp[i] := evalf(-b[i+1] - p*dbdp[i+1] - q*dbdp[i+2]):
            dbdq[i] := evalf(-p*dbdq[i+1] - q*dbdq[i+2] - b[i+2]):
        end do:
        r := evalf(a[1] - p*b[0] - q*b[1]):
        s := evalf(a[0] - q*b[0]):
        drdp := evalf(-b[0] - p*dbdp[0] - q*dbdp[1]):
        dsdp := evalf(-q*dbdp[0]):
        drdq := evaf(-p*dbdq[0] - q*dbdq[1] - b[1]):
        dsdq := evalf(-q*dbdq[0] - b[0]):
        printf("r = %e, s = %e\n", r,s);

        A := Matrix(2,2,[ [drdp,drdq], [dsdp,dsdq] ]):  #Using Maple LinearAlgebra
        C := Vector( [-r, -s]);
        det := A[1,1]*A[2,2] - A[1,2]*A[2,1]:
```

```
        if ( det <> 0 ) then
            h := LinearSolve(A,C):
            p:= p + h[1]:                          # update p and q
            q:= q + h[2]:

            hsize := sqrt( h[1]^2 + h[2]^2):       # test for convergence
            psize := sqrt( p^2 + q^2):
            if ( hsize/psize < htolerance ) then
                finish := 3:                       # OK, we have converged
            else
                j := j + 1:
                if (j > iterations) then
                    finish := 2:                   # Maximum iteration count reached
                end if:
            end if:
        else
            finish := 1:                           # the derivative matrix is singular
        end if:
    end do:
    if ( finish < 2 ) then
        printf("p = %e, q = %e, determinant zero, process aborted\n",p,q);
    elif (finish < 3) then
        printf("p = %e, q = %e, iterations exceeded, process aborted\n",p,q);
    else
        printf("p = %e, q = %e, converged in %d iterations\n", p,q,j);
    end if;
    return(finish):
end proc:

success := Bairstow(np, pA, pstart, qstart);
```