First name: Sebastian
Last name: Gumpold
Date: 14.11.03
Homework number: 1
Homework Title: Excercise 1.13

Problem description:

The Euclidean norm of an n-dimensional vector x is defined by

$$\|x\|_2 = \left(\sum_{i=0}^{n} x_i^2\right)^{\frac{1}{2}}$$

How would you avoid overflow and harmful underflow in this computation?

Problem solution:

An *overflow* could occur after adding an summand although the final result (after computing the square root) might not be too high for the computer.

A harmful *underflow* could occur when an element of the vector x is so small that the square of that number is smaller than the smallest number that can be computed, so the element does not have an influence to the result. That would be bad, if the input vector consists mostly of such small elements.

Results:

Overflow and underflow can be avoided by multiplying the input vector x by a scalar `scale`. This method is known as Scaling.

To avoid ***overflow*** this scalar should be the inverse of the maximal element of x:

```
scale := max(abs(x))            % get maximal element of |x|
ssq := 0
FOR i IN 1..n LOOP
     ssq := ssq + (x(i) / scale)^2
END LOOP
norm(x,2) := scale * sqrt(ssq)  % ||x||_2 := scale * ssq^(1/2)
```

It's necessary to divide each element of x by the maximal element of x (= `scale`) before squaring in order to avoid a potential overflow. To finally get the correct result `scale` is multiplied by the squareroot of the sum.

An ***underflow*** can be avoided in a similar way, by scaling with the minimal element of x.

```
scale := min(abs(x))            % get minimal element of |x|
ssq := 0
FOR i IN 1..n LOOP
     ssq := ssq + (x(i) / scale)^2
END LOOP
norm(x,2) := scale * sqrt(ssq)  % ||x||_2 := scale * ssq^(1/2)
```

In the case that there are very small and very large elements in the input vector at the same time this would be a possible algorithm to avoid ***both***:

```
scale := max(abs(x))                        % get max. element of |x|
ssq := 0
FOR i IN 1..n LOOP
     IF (abs(x(i)) >= sqrt(scale)*2)            % if overflow possible
          ssq := ssq + (abs(x(i)) / scale)^2    % scale with max element
     END                                        % else ignore element
END LOOP
norm(x,2) := scale * sqrt(ssq)        % ||x||_2 := scale * ssq^(1/2)
```

The algorithm above avoids possible overflows and approximates values, that have an absolute value smaller than `sqrt(max(abs(x)))*2` by 0 (in the concrete algorithm these elements are ignored) to avoid possible underflows. This can be done, because elements with an absolute value smaller than `sqrt(max(abs(x)))` wouldn't change the final result dramatically if there are also very large values in the input vector at the same time. In this way the algorithm avoids overflow and underflow.

Please note, that this algorithm only works properly if not only small elements (e.g. only values between -1 and 1) are present in the input vector.

Please also note, that I choosed a border of `sqrt(max(abs(x)))*2` to avoid an underflow when computing `abs(x(i))/max(abs(x))` and squaring the result.

If we would use a smaller border, e.g. `sqrt(max(abs(x)))` an underflow could happen (considering a machine precision of 2^-127 to 2^128) if `abs(x(i))` would be 2^64 and `max(abs(x))` is 2^128. This would result in (2^64 / 2^128)^2 = (2^-64)^2 = 2^-128, which is an value that would cause an underflow.

Discussion and Comments:

A problem of the solutions for avoiding overflow or underflow might be the complexity of the algorithms (it has to run two times through the whole input vector), which might result in a slow computation.