ANPASS is a program for performing least-squares analysis. It was originally written in Fortran but has since been translated to Go and then to Rust by Brent R. Westbrook. This documentation corresponds to the Rust version only. In principle, ANPASS can be used to solve any polynomial or even linear regression problem, but in practice it is only used for determining the force constants in quartic force fields. As such, this will be the running example of usage in this document.

1 Overview

A quartic force field (QFF) is a fourth-order Taylor series expansion of the internuclear potential energy portion of the Watson Hamiltonian of the form

$$V = \frac{1}{2}\sum_{ij} F_{ij}\Delta_i\Delta_j + \frac{1}{6}\sum_{ijk} F_{ijk}\Delta_i\Delta_j\Delta_k + \frac{1}{24}\sum_{ijkl} F_{ijkl}\Delta_i\Delta_j\Delta_k\Delta_l \tag{1}$$

This can be rephrased as the matrix equation $XF = V$, where F is the vector of force constants that minimizes the least squares difference $|XF - V|$, V is a vector of potential energies, and X is itself a matrix composed of the elements X_{ik} given by the expression

$$X_{ik} = \prod_j^N x_{ij}^{e_{jk}} \tag{2}$$

In turn, the x_{ij} for the QFF example are the jth components of the ith displacement, and the e_{jk} are the kth exponent in the jth row of exponents. This is similar to the basic polynomial regression problem given by

$$A = \begin{bmatrix} x_1^m & \dots & x_1^2 & x_1 & 1 \\ x_2^m & \dots & x_2^2 & x_2 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ x_n^m & \dots & x_n^2 & x_n & 1 \end{bmatrix} \tag{3}$$

except that $N = 1$ in the basic form, and k is $\begin{bmatrix} m & m-1 & \dots & 0 \end{bmatrix}$ instead of varying in arbitrary ways. In other words, the exponents are given by the vector $\begin{bmatrix} m & m-1 & \dots & 0 \end{bmatrix}$ instead of the matrix with elements e_{jk}, and the x_{ij} are really the vector of elements x_i.

With this formulation, ANPASS can be used to solve the mth-order regression problem. For example, a linear regression problem for i data points is represented by $e = \begin{bmatrix} 1 & 0 \end{bmatrix}$. The one caveat to this is that the resulting function is only printed in the file `fort.9903` in the units expected for a QFF, so if you want the raw values back, divide your coefficient by 4.359813653. A full example of such a fitting is given in Appendix A.

2 Input format

The layout of the ANPASS input file is as follows:

2.1 Header

```
!INPUT
TITLE
 H2O 2A1 F12-TZ
PRINT
   99
```

```
INDEPENDENT VARIABLES
   3
DATA POINTS
  69   -2
(3F12.8,f20.12)
```

The only line read by the Rust version is the last. This line is parsed by the regular expression:

```
"(?i)^\s*\((\d+)f[0-9.]+,f[0-9.]+\)\s*$
```

which looks for a line of this general form and also captures the first number before an `f`. This first number is used as the number of columns of displacements in the next section and signals the beginning of that section.

2.2 Displacements

This section is composed of displacements in the first $N-1$ columns, followed by an optional column of energies. If the number of columns minus one is equal to the number parsed in the aforementioned regular expression, the first $N-1$ columns are taken as displacements, while the last column is taken as the corresponding energy. Otherwise, all N columns are taken as part of the displacement matrix. This allows for only the displacements to be input as part of a template.

```
 -0.00500000 -0.00500000 -0.01000000      0.000128387078
 -0.00500000 -0.00500000  0.00000000      0.000027809414
 -0.00500000 -0.00500000  0.01000000      0.000128387078
 -0.00500000 -0.01000000  0.00000000      0.000035977201
 -0.00500000 -0.01500000  0.00000000      0.000048243883
 -0.00500000  0.00000000 -0.01000000      0.000124321064
 -0.00500000  0.00000000  0.00000000      0.000023720402
 -0.00500000  0.00000000  0.01000000      0.000124321065
 -0.00500000  0.00500000 -0.01000000      0.000124313373
```

2.3 Unknowns

This section simply lists the number of columns in the matrix that follows. This is necessary since the exponents are wrapped after 16 columns, so it would be possible for all of them to blend into a single row. This section looks like:

```
UNKNOWNS
  22
```

2.4 Exponents

This section gives the exponents for the polynomial equation to be fit. It has the form given in the listing below.

```
FUNCTION
   0    1    0    2    1    0    0    3    2    1    0    1    0    4    3    2
   1    0    2    1    0    0
   0    0    1    0    1    2    0    0    1    2    3    0    1    0    1    2
   3    4    0    1    2    0
   0    0    0    0    0    0    2    0    0    0    0    2    2    0    0    0
   0    0    2    2    2    4
```

2.5 Stationary point

This section requests a refitting of the energies to a known stationary point. It looks like:

```
STATIONARY POINT
     -0.000045311426     -0.000027076533      0.000000000000     -0.000000002131
```

Like the Displacement section above, the first $N-1$ columns are displacements used to bias the displacements in that section, while the last column is an energy used to bias each of the energies before performing the refitting. If this section is not present, a stationary point search is undertaken and a stationary point will be printed in the output.

3 Fitting

4 Stationary Point

A Linear regression example

The complete input for the linear regression example described in the Overview is shown below. The Fortran version fails to run on this example, and the Go version panics because it can't find a stationary point (since lines don't have stationary points). However, the Go version will still write `fort.9903`, which contains the coefficients of the line.

Listing 1: anpass.in

```
!INPUT
TITLE
 H2O 2A1 F12-TZ
PRINT
   99
INDEPENDENT VARIABLES
   3
DATA POINTS
  11   -2
(1F12.8,f20.12)
  0.00000000      2.300000000000
  1.00000000      3.400000000000
  2.00000000      7.600000000000
  3.00000000      8.100000000000
  4.00000000      9.400000000000
  5.00000000     13.600000000000
  6.00000000     14.500000000000
  7.00000000     15.900000000000
  8.00000000     18.600000000000
  9.00000000     21.700000000000
 10.00000000     21.800000000000
UNKNOWNS
   2
FUNCTION
   1    0
END OF DATA
!FIT
!STATIONARY POINT
!END
```

Listing 2: fort.9903

```
    1     0     0     0       8.894019852120
    0     0     0     0       9.789763384464
```

Again, the results found here have been converted to the units needed for a QFF, so dividing them both by 4.359813653 gives the desired equation $y = 2.04x + 2.24$.