A Tutorial for Using the ADAPT Software System For Structure-Property Relationship Studies

(Updated November 2, 2001)

The Data Set:

At times, methods commonly employed are best explained with an example. For those cases, a data set of 200 compounds will be assumed.

Getting Started

A study should be done in a new ADAPT data area. First, create a new directory and then execute the ADAPT routine, **AFINIT**. This will create all of the direct access binary files that are necessary to run the ADAPT software system. There should be 33 binary files as well as an *output* and an *input* file.

Entering the Molecular Structures

The HyperChem Molecular Modeling package is used to draw each of the compounds. Generally, optimizing structures in HyperChem before *FTP*ing them to a workstation reduces time spent on MOPAC optimizations. Save each compound on the hard drive as a *dan###.mol* file. For example, the first compound of the study should be saved as *dan001.mol*. (**HyperChem Lite users**: files will be saved as *dan###.hin*)

Worklist Generation

After all structures have been drawn and saved on a PC, *FTP* them as ASCII files to the ADAPT area directory on *hera* or *ares*. Input the structures into the ADAPT area using the routine **STRIN** (for *.mol) or **CONHIN** (for *.hin). Enter the range of *.mol files (or *.hin files) to process. For example, using the 200-member dummy data set, enter 1/200. When asked for the starting DAN #, enter 1.

CLSMKR

The routine **CLSMKR** (classmaker) inputs the structures as members of a worklist in the ADAPT area.

- *main* to specify the main descriptor area
- *inpu* to input the worklist
  - when prompted for how many classes to be made, enter 1
  - enter the range of DAN numbers for the worklist; example - 1/200
- *stor* to store the worklist
- *done* exits the program

VERY IMPORTANT NOTE: Once classmaker has been executed once, ***NEVER*** *run it again in that ADAPT area or it may cause severe problems with your study.*

Geometry Optimization

The script **MOPALL** can be used to optimize structures with MOPAC. When prompted, enter the *dan* files in the worklist to be optimized with MOPAC. Then, when prompted for the keywords, type: **PM3 T=99999.9 EF HESS=1 MMOK GNORM=**

- **PM3** invokes the PM3 Hamiltonian to be used during the optimization. We conventionally use the MNDO-PM3 because is has been shown to produce the most accurate geometries.
- **T** is a time limit on the optimization per structure.
- **EF** invokes the Eigenvector Following optimization procedure. Optimizations historically have been run using the BFGS quasi-Newton method, however, in recent years EF has been tested and shown to give equivalently accurate geometries, with shorter optimization times, and far fewer errors during the optimization. Its use it preferred now.
- **HESS** is used only in conjunction with EF. HESS=1 invokes the construction of the Hessian matrix prior to geometry optimization. This speeds up the process.
- **MMOK** invokes an increased barrier to rotation correction when a peptide bond is encountered.
- **GNORM** sets the gradient norm threshold for termination of an optimization. This is automatically calculated and appended to the end of the keyword string above.

These jobs may take a while, therefore put the job in the background (BE CAREFUL: if you background the script too soon, it will hang. Give it about a minute before you background). When the routine is finished, each structure should have a *.mol (or *.hin), *.arc. *.dat, and *.out file. Each structure needs to satisfy a geometry optimization criterion called "Peter's Test", (actually each structure needs to have either converged by a BFGS optimization or an SCF field must be achieved) which will be printed in each *.out file. You may search for the presence of this line by 3 methods:

1. Physically opening each of the *.out files and seeing if each contains the phrase "Peter's Test is Satisfied." If it does, the test has been satisfied and the optimization is complete.

2. All *.out files can be searched at once by typing **grep PETE *.out > output**. This will print a list of all DANs in which the sequence "PETE" appears to the output file.

3. Perhaps the best alternative is to use the script file '*pete*'. This resides in a variety of /bin directories, so just ask around where you can get this. One advantage of *pete* is that it checks for both Peter's Test and SCF Field Optimization and then writes all DANs that did not pass to a file called *pete.out*.

To "fix" structures that do not satisfy Peter's test, you can do three things:

- For some convergence problems, you can enter the DMAX keyword when running mopall and set its value to a lower number. By default, it is 0.2. Setting it to 0.1 or 0.05 can sometimes correct the problem.

- You can make small variations to bond lengths and angles in the *.dat files. Once these lengths have been modified, run **domop** followed by the DAN file in question (without a file extension). Example: *domop dan005*.

- The other option is to redraw the structures in HyperChem (Lite). If you choose this option, run **SFILES** and enter *dele* to delete the structure(s) in question. Also, delete the four files (.dat, .arc, .out, .mol (.hin)) for each of the DAN files in the working ADAPT area directory as well. After the structures have been redrawn, *ftp* them back into the ADAPT area and **STRIN** the structures back in the same way as before. Run **MOPALL** again only on the redrawn structures and check for optimization. This process may need to be repeated a couple of times for structurally unusual or large molecules.

- For some of the peskier error messages, consult with your favorite experienced group member for guidance about more specific troubleshooting procedures.

Once all structures have satisfied the optimization test, use **MOPOUT** or **AMOPOUT** to write the optimized structure coordinates into the ADAPT binary files. When prompted, enter all DANs in the worklist. Make sure you look at the **amopout.error** file. If coordinates were not replaced, try running **AMOPOUT.FORCE**. This should fix your problem. Use **MOLIN** (for HyperChem users) or **HININ** (for HyperChem Lite users) to write new *.mol or *.hin files, respectively. When prompted, enter all DAN files in the worklist. Next, *ftp* the files (in ASCII format) back onto the PC. Open each of the structures in HyperChem [or Lite] to make sure that all structures have been drawn and optimized correctly. If any structures are not optimized correctly, unusual bond lengths and angles will be readily apparent. If this occurs, redraw the structures after deleting them with **SFILES** as explained above. Once all structures are entered correctly, descriptor generation can be performed.

NOTE: You will need the re-run the geometry optimization using the AM1 Hamiltonian at some point as well to obtain some charge information (it is superior to PM3) for this purpose. In general, do this up front before anything further as well to save time later on. To avoid confusion, make a new directory within your study directory called AM1 and set up a new ADAPT area as described on page one of this tutorial. Copy all PM3 geometry-optimized structures to that directory and read them in, again, as described previously. Then, run MOPALL again, using AM1 instead of PM3 in the keyword list. Proceed as you did for the PM3 geometry optimization.

Training, Prediction and Cross-Validation Sets

Before descriptor generation and model building, separate the data set into sets – approximately 80% of the compounds will be put into a *training set*, 10% will be put into an external *prediction set* and 10% will be put into a *cross-validation set*. There are two ways to do this, with SETBIN and TSETS.

SETBIN

The program **SETBIN** generates sets pseudo-randomly by binning the range of experimental values (dependent variable) and then choosing observations from each bin. This ensures that a numerically representative sample of the data set is used for cross-validation and prediction.

- Create a file called depv.txt which contains the experimental values for all compounds in the study.
- Create a file called observations.txt which contains the *dan* file numbers corresponding to the values in depv.txt. A number of automated programs/scripts (**OBSERVE** or **OBS.LAN.PERL**) were written to handle this.
- Run SETBIN
    - enter the percentage of the data to be used for the PSET
    - enter the percentage of the data to be used for the CVSET
    - enter a random seed
- The file setbin.out contains the sets for bookkeeping purposes.
- The file tsets.in can be used with the program TSETS to enter the sets.
    - Type tsets < tsets.in

The sets will now be set up such that set 1 contains the combined TSET and CVSET compounds, set 2 will contain only the PSET compounds, set 3 will contain only the TSET compounds, and set 4 will contain only the CVSET compounds.

The longer, traditional way…

TSETS

The program **TSETS** will allow the completely random selection of compounds for formation of sets.

- *cgs* to generate computer generated random sets. When prompted, enter a random seed. You will be prompted to enter the number of training set members; enter the data-set size minus approximately 10%. For example, using the dummy data set of 200 members, the TSET would have 180 members and the PSET would have 20 members.. When asked how many sets are desired, enter 1. Both the TSET and PSET will be stored in set #1.
- *load* and enter *1*
- *disp* to list the compounds of both the TSET and PSET to the screen.
- *swst* to switch the members of the PSET and the TSET.
- *csets* to change the current working set to the PSET
- *wipe* to erase the PSET.
- *stor* and enter *2* to store this in the second storage space.
- *load* set *1* again.
- *csets* until you are working in the prediction set area

- *wipe* the structures.
- *stor* and enter *1*.

At this point, set 1 should be just your TSET and set 2 should be just your PSET.

To set up your CVSET:

- *load* and enter *1*.
- *cgs* to remove another random 10% of your structures (i.e. for the dummy example, enter a training set of 160).
- *stor* and enter *3*.
- *load* and enter *3* and perform the same routine as in the previous paragraph – only you will be setting up set 4 instead of set 2.

This will establish all sets that you will need for model development. Following this procedure you will have the following 4 sets:

Set 1: TSET (Training set w/o prediction set)
Set 2: PSET (Prediction set)
Set 3: TSET (Training set w/o prediction set & cross validation set)
Set 4: CVSET (Cross-validation set)

Setting up the Dependent Variable

Assuming you have the values for the property of interest in Excel, highlight and copy the list of data. Paste the values into the file *input* in the ADAPT area (make sure it is empty first). Execute the routine **CALC**.

CALC

- *finp* to read in formatted input from 'input' file
  - when prompted to enter up to 50 numbers, enter 1 (for LAN #1)
  - when asked to enter format, hit enter (for free-format)
  - when asked to enter a new label, enter "depv" (you can enter anything, but it is easy to remember that this stands for dependent variable)
  - don't enter a flag; hit enter
- *done* exits the program

Descriptor Generation

There are 27 ADAPT routines commonly used which calculate topological, electronic, geometric and combination descriptors. Because of ADAPT's heavy-atom limitations, Phil Mosier has re-written some topological descriptor routines to handle compounds containing up to 255 heavy

atoms. Descriptor calculation bypasses ADAPT and is done by reading the .mol files for all compounds. When running these routines, .mol files must be present in your ADAPT area. It would be in your best interest to consult with Phil before running these routines. They can be found on *ares* in **/disk1/users/pdm/ADAPT_PLUS/bin**. Descriptor routines that are ADAPT_PLUS-compatible will be marked with an asterisk(*). You can run all or most of these:

Topological	**Geometric**	**Electronic**	**Combination**
dkappa*	dmgeo	charge/pkachg	cpsa
dmalp	dmomi	dsc	hbpure/hbmix
dmchi*	savol	hleh	
dmcon*	shadow	dcarb	
dmfrag*	dgrav		
dmwp*	geowind		
ctypes*	loverb		
dedge*			
mpolr*			
mrfrac*			
dsym*			
destat*			
eccen*			
dpend*			

Common directives used for ADAPT routines

*DKAPPA**

The routine **DKAPPA** calculates the topological shape descriptors called kappa indices.

- *work* to calculate descriptors for the entire worklist
- *desc* followed by *0* to specify all 6 kappa descriptors
- *lans* to specify 6 LANS to store the descriptors
- *go* starts the calculation
- *done* exits the program

DMALP

The routine **DMALP** generates path descriptors.

- *work* to calculate descriptors for the worklist
- *go* starts the calculation
- *stor* to specify the 5 LANS for descriptor storage

  When asked if storing allp descriptors is ok, answer with "y" and enter the appropriate LANS
- *done* exits the program

*DMCHI**

The routine **DMCHI** generates molecular connectivity descriptors. (Use CAPS-LOCK!)

- *work* to calculate descriptors for the worklist
- *exec* to execute the program
- *stor* to choose which descriptors to store

valence (V) -- 19 LANS
- *stor* to choose which descriptors to store
  simple (S) -- 19 LANS
- *stor* to choose which descriptors to store
  counts (C) -- 20 LANS
- *done* exits the program

*DMCON**
The routine **DMCON** generates valence-corrected molecular connectivity descriptors.
- *work* to calculate descriptors for the worklist
- *go* to execute the program
- *stor* to choose which descriptors to store
- *done* exits the program

*DMFRAG**
The routine **DMFRAG** generates constitutional fragment descriptors.
- *work* to calculate descriptors for the worklist
- *go* to execute the program
- *stor* to choose which descriptors to store
  save all descriptors with non-zero values
- *done* exits the program

*DMWP**
The routine **DMWP** calculates weighted path descriptors.
- *work* to calculate descriptors for the worklist
- *desc* to enter the descriptors to calculate; enter *0* to specify all
- *lans* to specify 5 LANS to store the descriptors
- *go* starts the computation
- *done* exits the program

*CTYPES**
The routine **CTYPES** calculates the hybridization of carbon atoms based on connectivity only to other carbon atoms.
- *work* to calculate descriptors for the worklist
- *desc* to enter the descriptors to calculate (9 possible)
- *lans* to specify which 9 LANS for descriptor storage
- *go* starts the computation
- *done* exits the program

*DEDGE**
The routine **DEDGE** generates molecular distance-to-edge descriptors.
- *go* to execute the program
- *stor* to choose which descriptors to store
- *done* exits the program

*MPOLR**

The routine **MPOLR** generates a molecular polarizability descriptor.

- *work* to calculate descriptors for the worklist
- *stor* to store the descriptor in a LAN
- *go* to execute the program
- *done* exits the program

*MRFRAC**

The routine **MRFRAC** generates a molar refraction descriptor. (Use CAPS-LOCK!)

This is a conversational routine:

- "Calc mol. refr. for entire worklist?"
  YES
- "Now working on main descr area -- want to change?"
  NO
- "Want debug output?"
  NO
- "Store mref descr on disc?"
  YES
- "Enter LAN No. for storage"
  ###
- "Return to start?"
  NO

*DSYM**

The routine **DSYM** generates a structural symmetry descriptor.

- *work* to calculate descriptors for the worklist
- *meth* to set the search method
  topological
- *land* to specify a LAN for descriptor storage
- *go* to execute the program
- *done* exits the program

*DESTAT**

The routine **DESTAT** calculates electrotopological state index descriptors.

- *work* to calculate descriptors for the worklist
- *desc* to enter the descriptors to calculate (8 possible)
- *lans* to specify which LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

*ECCEN**

The routine **ECCEN** generates an eccentricity descriptor.

- *go* to execute the program

- *stor* to store descriptor in LAN (1 possible)
- *done* exits the program

*DPEND**

The routine **DPEND** generates the superpendentic index descriptors.

- *go* to execute the program
- *stor* to choose which descriptors to store in LANS (6 possible)
- *done* exits the program

DMGEO

The routine **DMGEO** calculates geometric moments of molecules (sort of)

- *work* to calculate descriptors for the worklist
- *desc* specify the descriptors to calculate; enter *0* for all
- *land* to specify which 6 LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

DMOMI

The routine **DMOMI** calculates moments of inertia (correctly)

- *work* to calculate descriptors for the worklist
- *desc* specify the descriptors to calculate; enter *0* for all
- *land* to specify which 7 LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

SAVOL

The routine **SAVOL** calculates the surface area and the volume of molecules.

- *work* to calculate descriptors for the worklist
- *land* to specify which LANS should be used for surface area and volume descriptor storage, respectively.
- *cpop* to set output; choose (4) for special output option (Clear 'INPUT' first!)
- *calc* to start the calculation
- *done* exits the program
- (Make sure to keep the file 'input')

SHADOW

The routine **SHADOW** calculates shadow area descriptors.

- *work* to calculate descriptors for the worklist
- *desc* to specify the descriptors to calculate (6 possible)
- *lans* to specify which LANS should be used for descriptor storage
- *ornt* to specify the orientation of the compound; selection option 2, to orient with the first two moments of inertia -- VERY IMPORTANT!
- *go* starts the computation
- *done* exits the program

DGRAV

The routine **DGRAV** generates gravitational index descriptors (heavy atoms only or hydrogens included).

- *work* to calculate descriptors for the worklist
- *desc* to specify which descriptors to calculate (9 possible)
- *lans* to specify which LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

GEOWIND

The routine **GEOWIND** calculates the 3-D Weiner Index descriptor.

- *wkls* to calculate descriptors for the worklist
- *mlan* to specify which LAN should be used for descriptor storage (1 possible)
- *go* starts the computation
- *done* exits the program

LOVERB

The routine **LOVERB** calculates the length-to-breadth ratio descriptor

- *work* to calculate descriptors for the worklist
- *desc* to specify which descriptors to calculate; select option 3 for both
- *lans* to specify which 2 LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

PKACHG

The routine **PKACHG** generates atomic charge and pKa descriptors.

- *work* to calculate descriptors for the worklist
- *desc* to enter the descriptors to calculate (5 possible)
- *lans* to specify which LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

** If problems arise with PKACHG, then run CHARGE.

CHARGE

The routine **CHARGE** generates atomic charge descriptors

- *work* to calculate descriptors for the worklist
- *desc* to enter the descriptors to calculate (4 possible)
- *lans* to specify which LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

** If problems arise with both PKACHG and CHARGE, then use CHARGE alternative:

- IN THE AM1 OPTIMIZATION DIRECTORY…

- Run **EXCHG2** to extract charge information from the MOPAC output files.
- Concatenate the files ‘exchg.out’ and ‘input’ to the file ‘surchg.in’.
- Copy ‘surchg.in’ to ‘input’
- Run **SURCHG** to create the files ‘surchg.out’ & ‘charge.desc’
- Copy ‘charge.desc’ to ‘input’ and edit file to remove all headers
- Copy ‘input’ to the PM3 optimization directory and run **CALC**
- *finp* to read in formatted input from ‘input’; specify 3 LANS
- name the descriptors successively: *qneg, qpos,* & *qsum*
- *done*
- grep ‘DIPOLE’ *.arc > dipole.out
- cut -f 2 -d = dipole.out | cut -f 1 -d D > temp
- mv temp input
- Copy ‘input’ to the PM3 optimization directory and run **CALC**
- *finp* to read in formatted input from ‘input’; specify 1 LAN
    - name the descriptor *dipo*
    - *done*

DSC

The routine **DSC** calculates sigma charge descriptors

- *work* to calculate descriptors for the worklist
- *desc* to enter the descriptors to calculate (3 possible)
- *land* to specify which LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

HLEH

The script **HLEH** extracts information about homo & lumo energies, electronegativity and hardness

- IN THE AM1 OPTIMIZATION DIRECTORY…

- *clean* ‘input’
- **HLEH** on all compounds to make file ‘hleh.out’
- copy ‘hleh.out’ to ‘input’
- Copy ‘input’ to the PM3 optimization directory and run **CALC**
- *finp* to read in formatted input from ‘input’; specify 4 LANS
- name the descriptors successively: *homo, lumo, elec & hard*
- *done*

DATOM

The routine **DATOM** calculates the average charge on cyano and carbonyl carbons. In addition, it calculates the following descriptors taking into account only nitrogen, oxygen, sulfur and halogen atom types individually.

- *load* to read the information from 'surchg.out'
- *desc* to enter the descriptors to calculate (22 possible)
- *mlan* to specify which LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

CPSA

The routine **CPSA** calculates charged partial surface area descriptors.

- *load* to read the information from 'surchg.out'
- *desc* to enter the descriptors to calculate (29 possible)
- *mlan* to specify which LANS should be used for descriptor storage
- *go* starts the computation
- *done* exits the program

HBPURE

The routine **HBPURE** calculates hydrogen bond specific descriptors for pure compounds.

** This program is generally used for physical property studies.

- *load* to read the information from 'surchg.out'
- *desc* to enter the descriptors to calculate (23 possible)
- *lans* to specify which LANS should be used for descriptor storage
- *chrg* to specify whether you used **charge** or **pkachg** for charge descriptors (if **charge** or **pkachg** did not work and you had to extract charges from MOPAC, tell the program you used **charge**)
- *go* starts the computation
- *done* exits the program

HBMIX

The routine **HBMIX** calculates hydrogen bond specific descriptors for mixed compounds.

** This program is generally used for biological activity studies.

- *load* to read the information from 'surchg.out'
- *desc* to enter the descriptors to calculate (23 possible)
- *lans* to specify which LANS should be used for descriptor storage
- *chrg* to specify whether you used **CHARGE** or **PKACHG** for charge descriptors (if **charge** or **pkachg** did not work and you had to extract charges from MOPAC, tell the program you used **charge**)

- *go* starts the computation
- *done* exits the program

Initial Descriptor Pool Reduction

At some point during the calculation of electronic or combination descriptors, the ADAPT limit of 200 LANS will be exceeded. To circumvent this issue, an initial reduction of the descriptor pool is performed using **DSCREEN** to remove enough descriptors to allow for calculation of rest.

DSCREEN
The routine **DSCREEN** finds descriptors which contain identical information and pairwise correlations.

- *wset* to enter the working set; make sure you only use the TSET!
- *itst* to specify the identical test cut-off percentage, typically *90*
- *rcut* to specify the R-value cutoff for pairwise correlations, typically *0.90* to *0.95*
- *depv* to specify the dependent variable LAN, enter *1*
- *add* to enter descriptor LANs, enter 2/### (where ### = last filled LAN)
- *seed* to enter a random seed
- *go* to execute the program
- *done* exits the program

For the initial descriptor pool reduction, only those descriptors that are identified as containing identical information should be removed using **DFILES**. This information can be obtained from the very end of the 'output' file.

DFILES
The routine **DFILES** is a descriptor maintenance program. Remove descriptors as follows:

- *dele* to remove descriptors by typing in the LAN numbers to delete
- *comp* to compress the ADAPT area; DO NOT reorder the descriptor numbers!
- *done* exits the program

Presently, the amount of descriptors that we can calculate exceeds the capabilities of **DSCREEN** (especially for fairly large data sets). Usually, once **DFILES** fills up, one has to run **DSCREEN**, remove identical descriptors, continue calculating descriptors, and then run **DSCREEN** again to get your final reduced pool. An alternate way to perform objective feature selection is to use the program **CORREL,** which resides in many group members' bins. CORREL can handle up to 1000 descriptors and 1000 compounds. It works in the exact same way as **DSCREEN** does.

Objective Feature Selection

The routines **DSCREEN** and **VSDA** are used for the purpose of feature selection. The overall pool of descriptors is reduced by three methods of objective feature selection to generate a subset of the most information rich descriptors, without the use of the dependent variable.

- The first is identical testing, which is used to remove any descriptor with greater than 90% identical values. This is important because not much useful information would be obtained from a particular descriptor if each value of that descriptor were identical for most of the compounds in the data set. For example, if each structure had two chlorine atoms, the topological descriptor, number of chlorine atoms, would not be useful.

- The second is pairwise correlation, which is used to remove one of two descriptors that provide very similar information. For example, if the electronic descriptors, charge on the most positive atom and charge on the most negative atom varied from compound to compound, but the difference between their values was relatively constant across the data set, then only one of the two would be useful since one is just a scalar of the other. Keeping descriptors with similar information in the final reduced pool would be redundant because the key is to obtain different information.

- The third attempts to maximize the mutual orthogonality among the descriptors. If you consider a descriptor as a vector, then two descriptors that contain no redundant information should be completely orthogonal to one another. By maximizing the orthogonality of the descriptors, you maximize the information content while lowering the number of descriptors in the reduced pool.

Subjective feature reduction is also applied in a QSPR study. In this type of procedure, the dependent variable is used to find subsets of descriptors that correlate best with the physical property being studied.

Now that all possible descriptors have been calculated and **DSCREEN** has been used to reduce the pool following the above instructions, a file named ‘dscreen.out’ which lists all of the descriptors in the final reduced pool should be present. At this point, the first thing to examine is the number of descriptors that have passed objective feature selection. A major criterion of model development is that the ratio of descriptors reduced descriptor pool to number of observations (# of compounds) be less than or equal to 0.6. If your descriptor pool out of **dscreen** is too large, you may submit the descriptors to **vsda**.

VSDA

The routine **VSDA** performs feature selection by eliminating weakly orthogonal descriptors

- clean ‘output’ file before starting program

- *wset* to specify the TSET
- *add* descriptors of the reduced pool from ‘dscreen.out’ file
- *load* descriptor information
- *plim* to set projection angle limit; change to 0.01
- *orth* to specify a basis vector to check orthogonality against
- *step* to perform orthogonalization

- *clbv* to clear the present basis vector

Generally, orthogonalizing against 5 basis descriptors is sufficient. The results from each run will be appended to the ‘output’ file.

At this point, FTP the ‘output’ file to a PC. Determine the maximum number of descriptors allowable using the aforementioned ratio. Then, in Excel, make a column of each of the 5 **vsda** runs using only the top maximum number of allowable descriptors. For example, a smaller data set may only be able to accommodate 30 descriptors. You would make 5 columns of the 30 most orthogonal descriptors from each run. Then, the easiest thing to do is to sort by ascending or descending order and find descriptors that appear in at least 4 of the 5 columns. Once you get the final reduced pool, remove the previous descriptors from the ‘dscreen.out’ file and copy and paste the new ones in their place.

Type I Model (Linear Feature Selection/Linear Model Development)

The programs **genlin** and **annlin** produce type I models by genetic algorithm and simulated annealing, respectively. The 'dscreen.out' file is set up to make a table of descriptor values to be read into either program by typing *nettab < dscreen.out*. This writes a file called 'genlin.in'. If annlin is being used, copy the 'genlin.in' file to 'annlin.in'.

GENLIN

The routine **GENLIN** finds information-rich subsets of descriptors using a genetic algorithm.

- *load* to read in data from 'genlin.in'; affirm that LAN 1 is the dependent variable
- *seed* to enter a random seed
- *tval* to set the internal validation T-value; enter 4
- *init* to initialize the population string; enter 50 population strings, length of string is the number of descriptors in the model…begin with 3 and work up.
- *iter* to enter the number of iterations; enter 1000.
- *grun* to execute the program

ANNLIN

The routine **ANNLIN** finds information-rich subsets of descriptors using simulated annealing.

- *load* to read in data from 'annlin.in'; affirm that LAN 1 is the dependent variable
- *seed* to enter a random seed
- *vali* to set the internal validation T-value; enter 4
- *desc* to enter the number of descriptors to include in the model; start with 3
- *arun* to execute the program

** Continue running genlin or annlin while increasing the number of descriptors in the model. Modeling up to 10-descriptors is generally sufficient. Each successive run will be appended to the respective *.out file. When you have finished building models, examine the *.out to find the best size model. The ideal size model is one with as few descriptors as possible with as low an rms value as possible.

** If no models are found with T-values greater than 4, then try lowering the validation value to 3.5 or 3. If all else fails, don't issue the *vali* directive at all.

** Once the size of the best descriptor subset has been determined, all ten models given by genlin or annlin can be examined to see which is best.

The scripts ANNVAL and ANNVALR automate the process of model searching for ANNLIN. When you are able to confidently do everything manually with ANNLIN and GENLIN, then you can use these scripts to make life a little easier.

Descriptor Subset Analysis

IRA

The routine **IRA** (interactive regression analysis) calculates model coefficients, standard error of the coefficients, T-values (the greater in magnitude the better), P-values (the smaller the better), and overall F-value (the greater the value the better).

- *tset 1 (stored tset #)* to enter the number in which the tset is saved
- *inpu 1 d1 d2 d3 d4…* to input the descriptors for the model under investigation
- *regr 1 all* to regress all model descriptors against the dependent variable
- *smod m1* to store current model in a MAN
- *done* exits the program

Other useful **IRA** directives:

- *add* to manually add descriptors to a model
- *drop* to manually remove descriptors from a model
- *cpop* to change the printing option; defaults to no output.

Testing for Compound Outliers

DDG

The routine **DDG** (data diagnostics generation) is used to examine a saved model for the presence of outliers by various regression analyses. Seven diagnostics are calculated, and a compound that fails four of these tests is considered an outlier. The diagnostics include residual, standardized residual, studentized residual, leverage value, DFFITS, Cook's distance, and Mahalanobis distance.

- *modl* to load the stored model of descriptors
- *depv* to define the LAN number of the dependent variable
- *wset* to run diagnostics using the training set only
- *rdia* to compute regression diagnostics for outlier detection
- *prin* to print the information to the 'output' file
- *done* exits the program

View the 'output' file and examine the last 7 columns of values. Values marked with asterisks (*) are outlier values. Generally, compounds with 4 or more asterisks are considered outliers.

Checking for Multicollinearities

COLATE
The routine **COLATE** preprocesses data for use with the routine **MLRA**.

- *wipe* to clear previous stored descriptors
- *real* to store real numbers instead of integers
- *qsar* to enter QSAR mode
- *add* to add the model descriptors
- *stor* to store the list of descriptors
- *done* exits the program

MLRA
The routine **MLRA** is used to check the variance inflation factor (VIF), which tests for the presence of multicollinearities in your model. Generally, you would like to see $r^2 < 0.90$ when a descriptor is regressed against all other descriptors in the model.

- *load* to load collated data set; answer yes to print out the correlation matrix
- *tset* to specify the training set
- *regr* to perform the regression
- *done* to exit the program

Examine the 'output' file. Under the column header "MCC/ADJ", check to see that no values are above 0.95. If so, examine other models to see if a better scenario exists.

Linear Regression Prediction

LRPRED
The routine **LRPRED** performs predictions of the property for the TSET and PSET.

- *modl* to load model
- *mdes* to load descriptors associated with the model
- *wset* to specify the TSET
- *go* to perform prediction
- *comp* to compare predicted values to dependent variable; enter 1
- *wset* to specify the PSET
- *go* to perform prediction
- *comp* to compare predicted values to dependent variable; enter 1

Information will be written to the 'output' file. FTP this to a PC, open it up in Excel and plot it in Sigma Plot.

Type II Model (Linear Feature Selection/Non-linear Model Development)

A type II study is done using the descriptors from the best linear type I model, but applying it to a computational neural network (CNN). The result is a linear/non-linear hybrid model.

NETTAB
The program **NETTAB** is used to generate a descriptor table for use with CNNs.

- *make* to create a table
  - Working set: Set 3 (TSET)
  - Qnetin format
  - Enter the descriptor LANs from Type I Model and the dependent variable last
  - Name the file 'table3'

- *make* to create a table
  - Working set: Set 2 (PSET)
  - Qnetin format
  - Enter the descriptor LANs from Type I Model and the dependent variable last
  - Name the file 'table2'

- *make* to create a table
  - Working set: Set 4 (CVSET)
  - Qnetin format
  - Enter the descriptor LANs from Type I Model and the dependent variable last
  - Name the file 'table4'

cat table3 table2 table4 > qnetin.pat

There are two programs that are used for Type II Model development, **QNET** and **ANN**. **QNET** uses the genetic algorithm and **ANN** uses simulated annealing to train the model. Additionally, there is a script called **QAUTO** that automates multiple qnet runs starting from different seed values. Before any of these programs may be run, the program **QNETIN** needs to be run to create initialization files.

QNETIN
The program **QNETIN** creates initialization files for running **ANN** and **QNET**.

- *train* to specify the training mode
  - ndump every 5-10 cycles
  - enter the name 'cycle' for weight file names
  - train for 1000 cycles; up to 2500 if necessary
- *layer* to specify the neural net architecture; enter # of input, hidden & output neurons
- *patterns* to specify the TSET, PSET and CVSET; writes 'qnet.pat' file
  - enter "n" to generate sets randomly
  - when asked to enter members of the prediction set, a very easy method to follow is:

    Assign the TSET, PSET and CVSET in sequential order. That is to say, assuming a 200-member data set, the TSET would be compounds 1/160, the PSET would be 161/180, and the CVSET would be 181/200.

  - the same goes when prompted to enter members of the CVSET
  - the TSET will automatically be assigned
- *trans* to input CNN neurons to be transformed; enter the input and output neurons
- *weight* to enter a random seed; writes 'qnet.wts' file
- *mode* to change from interactive to non-interactive mode
- *write* to write the 'qnet.in' file
- *done* exits the program

QNETIN is the program that establishes the starting seed for training the model. Because of this, it should be run for each individual **QNET** or **ANN** run that is performed, of course using a different seed each time. Generally, it is a good idea to perform at least 3-5 ann runs as well as 3-5 qnet runs OR qauto. Remember though, when writing new qnet.* files with **QNETIN** you need to copy the previous files to a directory where a particular ann or qnet run will be performed. All-in-all, several sets of files will be written and care must be taken to ensure that a good file system is set up to easily keep track of where the files are being copied. In any event, the procedure to run **QNETIN** multiple times is a bit simpler if an old qnet.in file exists in the main ADAPT area:

- *useold* to load information from existing 'qnet.in'
- *patterns* to specify the TSET, PSET and CVSET; writes new 'qnet.pat' file
- *weight* to enter a random seed; writes new 'qnet.wts' file
- *write* to write the 'qnet.in' file
- *done* exits the program

Once all files are where they should be, ann and qnet or qauto may be run.

ANN

The program **ANN** uses a generalized simulated annealing optimization technique to get a good starting set of weights and biases.

- load to input data from 'qnet.in'

- cycl to specify the number of elapsed cycles before checking for optimality; 50
- cost ; a good first approximation is to use the Type I Model TSET rms
- cvfc to enter the cross-validation factor; enter 0.4 or 0.5.
- seed to enter a random seed
- arun to execute the program

** Depending on the architecture, this may take a minute or so. Put in the background.

** Examine the 'ann.out' file. The farthest right column of values should be the CVSET rms error. Find the lowest error value and note the 'cycle' file. Copy that 'cycle.wts' file to 'qnet.wts'. Also, edit the 'qnet.in' file and change the word "TRAIN" to "TEST". Run QNET. This will take no more than 5-10 seconds. The file 'qnet.out' will be written which contains the prediction values and rms errors of the TSET, CVSET and PSET.

QAUTO

The program **QAUTO** performs multiple qnet training runs using different starting seeds.

- Enter the number of training sessions you would like run
- Specify the number of elapsed cycles before checking for optimality; 50
- Enter the first seed

** This takes a while; place it in the background.

** After the program is finished, view the 'qauto.out' file. This contains information for the best run with each differing seed. It contains: the seed used, the training session in which the optimal CVSET error was found, and the 'cycle' file which contains the optimum weights. Take note of the seed, and then run **QNETIN** to input that seed as the starting weight. Run **QNET** again and view the 'qnet.out' file. This will contain similar information as seen in the 'ann.out' file. Again, look for the minimum CVSET rms error and then test the network as detailed above for ann.

If one so chooses, instead of running **QAUTO** which automates multiple qnet runs, QNET itself can be run a few times. This is merely done be doing the same TRAINing and TESTing as detailed above for **QAUTO**.

Various group members that automate the process of Type II methodology wrote several scripts. In particular, using a committee of several ann runs to average network output has proved beneficial. When you are able to confidently do everything manually with QNETIN, ANN, QNET, or QAUTO, then you can use these scripts to make life a little easier. When that time comes, find your favorite group member to discuss possibilities for the automation process.

Type III Model (Non-linear Feature Selection/Non-linear Model Development)

Type III models are the most computationally intensive of the models to develop. The main difference between Type III and the Type I and Type II models is that non-linear feature selection is used to find information-rich subsets of descriptors. This is done by submitting the reduced pool of descriptors from dscreen and vsda to **GENDES** or **GENDESP**.

- Edit the 'dscreen.out' file to look something like this:

make

⇐ specifies to make a table

3

⇐ specify the TSET only (w/o PSET & CVSET)

2

⇐ specify gendes(p) format

d1

⇐ descriptor 1

d2

⇐ descriptor 2

d3

⇐ descriptor 3

d4

⇐ descriptor 4

.

and so on

.

and so on

.

and so on (of reduced pool ONLY!)

gendes(p).in

⇐ specify the file name

done

⇐ specify to exit the program

Then, *nettab* < *dscreen.out* ⇒ writes ‘gendes(p).in’

GENDES

- *load* to read information from gendes.in
  - Do not enter any prediction or cross-validation set members; hit enter
  - No exclusion set members; hit enter
  - Do no use a previous population; enter “n”
- *qnet* to set network parameters
  - set network layers; most likely 3
  - specify # of input, hidden and output neurons
  - train for about 1000 cycles
- *ann* to initialize simulated annealing parameters
  - enter the cost from your best ann or qnet run
  - enter the minimum cycle to check for optimality; enter 50
  - enter the cv factor; usually 0.4 or 0.5
- *gene* to initialize genetic algorithm parameters
  - enter the population strings; enter anywhere from 40 to 50
  - set the genetic algorithm iterations; a 1000 is sufficient
  - *go* executes the program

** Place job in the background (*CTRL-Z*, type *bg*) and *nice* it. (*renice 15 PID#*)

** The completed job will write ‘gendes.out’. This will contain the 10 best models found and their corresponding cost functions. Evaluate these models the exact same way as in a Type II Model, using **ANN** and **QNET** or **QAUTO**.

GENDESP

- *load* to read information from gendesp.in
  - enter the fraction left out for the CVSET; enter 0.2
  - Do not enter any prediction or cross-validation set members; hit enter
  - Do no use a previous population; enter “n”
- *qnet* to set network parameters
  - set network layers; most likely 3
  - specify # of input, hidden and output neurons
  - train for about 1000 cycles
- *ann* to initialize simulated annealing parameters
  - enter the cost from your best ann or qnet run
  - enter the minimum cycle to check for optimality; enter 50
  - enter the cv factor; usually 0.4 or 0.5

- *gene* to initialize genetic algorithm parameters
  - enter the population strings; enter anywhere from 40 to 50
  - set the genetic algorithm iterations; a 1000 is sufficient
  - *go* executes the program

** Place job in the background (*CTRL-Z*, type *bg*) and *nice* it. (*renice 15 PID#*)

** The completed job will write 'gendesp.out'. This will contain the 10 best models found and their corresponding cost functions. Evaluate these models the exact same way as in a Type II Model, using **ANN** and **QNET** or **QAUTO**.

A Note on Type III Models

For long jobs, examining the gendes(p).out file periodically to see if the rms error has converged or not can help conserve CPU time. If you wish to stop a gendes(p) job gracefully (with the best models printed to the output file), delete the gendes(p).run file and let the job terminate itself.

Monte Carlo Randomization Experiments

A further way to prove the validity of our models is to perform Monte Carlo randomization experiments. The dependent variable is randomly scrambled and used during training to develop models using the methods described previously. If the models developed using the actual dependent variable are true structure-property relationships, then models built with the scrambled dependent variable should be very poor in terms of the rms error and the correlation coefficient (R^2). This is because no relationship should exist between the structures and the scrambled dependent variable. Good models found with the scrambled dependent variable is evidence that the original model may have contained chance correlations between descriptors and the property or activity of interest.

- *clean* 'output'
- In **DFILES**, make a simple table of the dependent variable LAN for all the compounds in the data set.
    - *tabl* to make a table of descriptor values**,** *S* for simple table, *0* for entire work list
    - enter the LAN ### containing the dependent variable
- A sequential list of dependent variable values for the data set will be written to the file 'output'.
- *Edit* 'output' and remove all lines not containing depv values (i.e. remove the ADAPT output headers in the file and extra spaces at the end of the file).
- *Copy* 'output' to 'scramble.in'.
- Run **SCRAMBLE**. This program resides in various group members' bin directories.
- This should produce a 'scramble.out' file containing your scrambled dependent variables.
- In your ADAPT area, *copy* 'scramble.out' to 'input'.
- Run **calc**, *finp*, etc… to enter your random dependent variable into dfiles. Name the LAN accordingly.

Type I Randomization Experiments

While performing Monte Carlo experiments, good housekeeping is recommended so original files for your study are not lost or copied over.

- *Copy* 'dscreen.out' file (generated during Type I model formation) to your ADAPT area.
- *Edit* 'dscreen.out' to replace the dependent variable found at the end of the sequential list of LANs for the reduced pool with the LAN containing the scrambled depv.
- Issue the command, *nettab < dscreen.out*.
- Input ('annlin.in' or 'genlin.in') file is made which has all the descriptor values for compounds in your training set with the random dependent variable being the last descriptor.
- With this new input file, type I modeling can be performed as previously described to find the best subset of descriptors that correlates structure to the scrambled depv.
    - **Note**: When choosing subset size in **GENLIN**, **ANNLIN** or **ANNVALR**, only use the subset size that gave you the best results with your real depv. For

example, if your best type I model happened to be a five-descriptor model, only explore five-descriptor subsets with your scrambled depv. Furthermore, you want validation to be off when modeling with the scrambled depv (i.e. when issuing *valid* command in **ANNLIN,** type *0* for no validation).

- After running **GENLIN**, **ANNLIN** or **ANNVALR**, look at the output ('annlin.out' or 'genlin.out') file to find the best subset of descriptors.
- From there use **IRA** to enter your model into ADAPT. Bypass **DDG**, **COLATE**, and **MLRA**. Just run **LRPRED** to obtain your predictions for your training and prediction sets.
- Make a plot of calculated vs. experimental values for your data set. Type I Monte Carlo plots should appear like a shot-gun was used to blast your compounds on the plot or their should be a cluster of compounds right near the average-line.

Type III Randomization Experiments

- *Copy* 'dscreen.out' file (used for generation of 'gendes(p).in' file) to your ADAPT area.
- *Edit* 'dscreen.out' to replace the dependent variable found at the end of the sequential list of LANs for the reduced pool with the LAN containing the scrambled depv.
- *nettab < dscreen.out*
- 'gendes(p).in' file is made as before except your scrambled dependent variable being the last descriptor.
- With this new 'gendes(p).in' file, run **GENDES(P)** with the same CNN architecture used to find your best Type III model with the real depv. For example, if you used a 10-3-1 CNN architecture in searching your reduced descriptor pool to find optimal models, use that same architecture for the randomization experiment.
    - **Note**: After **GENDES(P)** is done running, best models will be listed at the end of the gendes(p).out file. Examine best model using Type II methodology used above with one exception. Only test the CNN architecture that gave the best model in the original QSAR study. For example, if the best model was found using a 10-6-1 CNN architecture, use that same architecture when examining your best model found with your scrambled depv.
- Make a plot of calculated vs. experimental values for your data set. Type III Monte Carlo plots should cluster your compounds right near the average-line.