ADDENDUM V: *PERL SCRIPTS*

- *Angle_strain.pl*

```
#! /usr/local/bin/perl
# Last update: nov/98, Renata Kover
# angle_strain.pl
# evaluates angle_strain_average in the molecule
# taking into account the hybridization of the atom. The output is a
# script to be used with pdb files for raswin.
# usage: angle_strain.pl inputfile(.log) scaler

# error checking
if ($ARGV[0] eq "") {
        print "\nusage: angle_strain.pl inputfile scaler (default 5)\n";
        print "\n Evaluates angle_strain_average in the molecule from a gaussian \n";
        print " log file taking into account the hybridization of the atom. The output is
a \n";
        print " script to be used with pdb files for raswin.\n \n";
        exit;
}

# Open the log file for reading
$file = shift; $scaler = shift;

open (IN, "< $file") or die "Where is $file?\n";

# First, build the atom number-atom type conversion
$get_names = "";
$line_num = 0;
$input_orientations = 0;
while (<IN>) {
        # Should we begin reading in the atom center/number info?
        if( (/Input orientation/) and ($input_orientations == 0) ) {
                $get_names = "true";
        }

        if( $get_names eq "true" ) {
                ++$line_num;
                next if( $line_num < 6 );
                if( /---------/ ) {
                        $get_names = "false";
                        ++$input_orientations;
                        next;
                }
                ( $center_num, $atom_num ) = split;
                $atom_nums[$center_num] = $atom_num;
                # print "center_num=$center_num atom_num=$atom_num\n";
        }
}

# DEBUG PRINT
# foreach $atom_num ( @atom_nums ) {
#       print "atom_num[$count]=$atom_num\n";
#       ++$count;
# }

# Look in a PERL book and learn how to REWIND a file
```

```
close( IN );
open (IN, "< $file") or die "Where is $file?\n";

while (<IN>) {
        # Begin looking for the angle info after these words
        if (/ Optimized Parameters /) { $opt="t"; }
        next unless ( $opt );

        # Angle info looks like this
        # A1    A(2,1,3)             110.0984         -DE/DX =    0.
        next unless (/\! A/);
        # print;

        # get the desired values out of the file
        if (/ \! A\d+\s+A\((\d+)\,(\d+)\,(\d+)\)\s+(\S+)/) {
                ($a, $b, $c, $deg) = ($1, $2, $3, $4, $5);
                $angle = "A($a,$b,$c)";

                # This is a 2D assoc. array
                $angle{$b}{"$a $c"} = $deg;
                # print "$angle $deg \n";
        }

}
close (IN);

# Grab the atom names
@atoms_mid = sort bynum keys %angle;
#print @atoms_mid, "\n";


foreach $atom_mid ( @atoms_mid ) {
        @atoms_other = keys %{ $angle{$atom_mid} };

        # print "There are ",$#atoms_other+1," angles involving atom $atom_mid\n";

        # If there are 1 or 6 angles then it is sp3 hybridization
        if ( ( $#atoms_other+1 == 6 ) or ( $#atoms_other+1 == 1 ) ) { $hyb{$atom_mid} =
"sp3"; }
        # Otherwise it is sp2 hybridization
        else { $hyb{$atom_mid} = "sp2"; }

        # loop through each angle and print out a report
        $num_angles = $#atoms_other+1;
        # print "Found $num_angles angle on atom number $atom_mid, the hybridization is
$hyb{$atom_mid}\n";

        foreach $atom_other ( @atoms_other ) {

                # Figure out the explicit names of the other atoms
                ($a, $c) = split " ", $atom_other;

                # sp2 carbon atom
                if ( ($hyb{$atom_mid} eq "sp2") && ($atom_nums[$atom_mid]==6) ) {
                        # print "Calculating the offset for an sp2 carbon\n";
                        &calc_offset( 120 );

                # sp3 carbon atom
                } elsif ( ($hyb{$atom_mid} eq "sp3") && ($atom_nums[$atom_mid]==6) ) {
                        # print "Calculating the offset for an sp3 carbon\n";
                        &calc_offset( 109.6 );

                # sp3 oxygen atom
                } elsif ( ($hyb{$atom_mid} eq "sp3") and ($atom_nums[$atom_mid]==8) ) {
                        # print "Calculating the offset for an sp3 oxygen\n";
                        &calc_offset( 109.6 );

                # sp3 sulfur atom
                } elsif ( ($hyb{$atom_mid} eq "sp3") and ($atom_nums[$atom_mid]==16) ) {
                        # print "Calculating the offset for an sp3 sulfur\n";
                        &calc_offset( 92 );

                } else {
```

```
                        print "Whoa there pardner, I don't understand the hybridization of
                        atom $atom_mid\n";
                        print "atom_num=$atom_nums[$atom_mid]\n";
                        exit;
                }

                # print "    found an offset of $offset\n";

                $angle = "A($a,$atom_mid,$c)";
                # printf ("%15s %4s %8.4f %6.2f\n",
                #       $angle, $hyb{$atom_mid}, $angle{$atom_mid}{$atom_other}, $offset );

                # Store the total percentage strain for each mid atom
                $strain{$atom_mid} += $offset;
        }
        # Normalize for the number of angles involved in the strain
        # calculation of each atom
        $strain{$atom_mid} *= 100/$num_angles;

}

# Print a rasmol input script

print "background white\n";
print "select all\n";
print "color label blue\n";
print "label on\n";
print "wireframe 20\n";
print "color bond [155,155,155]\n";

foreach $num ( @atoms_mid ) {
        $size = $strain{$num} * $scaler;
        $hybridization = $hyb{$num};

        print "#####################\n";
        print "# Below is for atom number $num with hybridization $hybridization\n";
        print "# and total strain percentage $strain{$num}\n";
        print "select atomno=$num\n";
        printf ("spacefill %1d\n", $size);
        print "colour red\n";
        # print "colour blue\n" if ($charge >= 0);
        print "\n";
}

sub calc_offset {
        $ideal = shift;

        # Calculate the normalized deflection from the ideal sp3 angle
        $offset = ( $ideal - $angle{$atom_mid}{$atom_other} ) / $ideal;

        # Square this offset value
        $offset *= $offset;
}

sub bynum { $a <=> $b };
```

- *Get_charge.pl*

```
#! /usr/local/bin/perl
# Last update: april/98, Renata Kover
# get_charge.pl
# requires programs:
# evaluates charges in the molecule and writes a script
# to be used with pdb files.
# usage: get_charge.pl scaler <inputfile(.log)

# error checking
if ($ARGV[0] eq "") {
```

```
        print "usage:\n get_charge.pl scaler (default 100) < input file\n";
        exit;
}

$scaler=shift;

while (<>) {

        if (/Optimization completed/) {
                $continue = "t";
        }

        next unless ( $continue );

        # Okay, we are past the "Opt. completed" line

        if (/Total atomic charges/) {
                # Save charge info after the
                # "Total atomic charges" line
                $get_charge = "t";
        }

        if (/Sum of Mulliken charges/) {
                # When we hit the "Sum of..." line,
                # stop getting the charge info
                $get_charge ="";
        }

        next unless ( $get_charge );

        ++$count;
        next unless ( $count > 2 );
        # Count how many lines after "Total atomic..."
        # we are.  Keep only AFTER line 2

        #################
        # The good stuff
        #################

        # Grab the atom number and type
        ( $a, $num, $type, $charge ) = split (/\s+/, $_);

        # print "The charge on atom num $num with type $type is $charge\n";

        $charges[$num] = $charge;

}

# Print a rasmol input script

print "background [200,200,170]\n";
print "select all\n";
print "color label blue\n";
print "label on\n";
print "wireframe 20\n";
print "color bond [155,155,155]\n\n";

foreach $num ( 1 .. $#charges ) {
        $charge = $charges[$num];
        $size = abs( $charge * $scaler );

        print "######################\n";
        print "# Below is for atom number $num with charge $charge\n";
        print "select atomno=$num\n";
        printf ("spacefill %1d\n", $size);
        print "colour red\n" if ($charge < 0);
        print "colour blue\n" if ($charge >= 0);
        print "\n";
}
```

- *Get_spin.pl*

```
#! /usr/local/bin/perl
# Last update: april/98, Renata Kover
# get_spin.pl
# requires programs:
# evaluates spin in the molecule and writes a script
# to be used with pdb files.
# usage: get_spin.pl scaler <inputfile(.log)

# error checking
if ($ARGV[0] eq "") {
        print "usage:\n get_spin.pl scaler (default 100) < input file\n";
        exit;
}

$scaler=shift;

while (<>) {

        if (/The wavefunction is already stable/) {
                $continue = "t";
        }

        next unless ( $continue );

        # Okay, we are past the "Opt. completed" line

        if (/Total atomic spin densities/) {
                # Save charge info after the
                # "Total atomic charges" line
                $get_charge = "t";
        }

        if (/Sum of Mulliken spin densities/) {
                # When we hit the "Sum of..." line,
                # stop getting the charge info
                $get_charge ="";
        }

        next unless ( $get_charge );

        ++$count;

        next unless ( $count > 2 );
        # Count how many lines after "Total atomic..."
        # we are.  Keep only AFTER line 2


        #################
        # The good stuff
        #################

        # Grab the atom number and type
        ( $a, $num, $type, $charge ) = split (/\s+/, $_);

        # print "The charge on atom num $num with type $type is $charge\n";

        $charges[$num] = $charge;

}

# Print a rasmol input script

print "background [170,200,200]\n";
print "select all\n";
print "color label blue\n";
print "label on\n";
print "wireframe 20\n";
print "color bond [155,155,155]\n\n";
```

```
foreach $num (1 .. $#charges) {
        $charge = $charges[$num];
        $size = abs( $charge * $scaler );

        print "######################\n";
        print "# Below is for atom number $num with charge $charge\n";
        print "select atomno=$num\n";
        printf ("spacefill %1d\n", $size);
        print "colour red\n" if ($charge < 0);
        print "colour blue\n" if ($charge >= 0);
        print "\n";
}
```

- *Gaussian2pdb.pl*

```
#! /usr/local/bin/perl
# Last update: Mar/98, Renata Kover
# gaussian2pdb.pl
# requires programs: zmatrix_extract, newzmat and pdbclean.pl
# Converts gaussian ts log files into individual pdb files
# usage: gaussian2pdb.pl inputfile(.log)

# error checking
if ($ARGV[0] eq "") {
        print "usage: gaussian2pdb.pl inputfile(.log)\n";
        exit;
}

$file = shift;

# Keep the word to the left of the period in the filename
$file =~ s/^(\w+).*/$1/;

`zmatrix_extract < $file.log > $file.temp`;

open (IN, "< $file.temp") or die "Where is $file.temp\n";
open (OUT, "> $file.out");

while (<IN>) {

        if ($count > 1) { print OUT; }

        if ( /^#/ ) {
                # This is only true the first time
                next if ( ++$count == 1 );
                print OUT;
                ++$count;
                next;
        }

}

close IN;
close OUT;

`newzmat -izmat -obkv $file.out`;
`pdbclean.pl < $file.bkv > $file.pdb`;

unlink "$file.temp";
unlink "$file.out";
unlink "$file.bkv";


exit;
```

- *Zmatrixextract*

```
#! /usr/bin/awk -f
# @(#) extract Gaussian zmatrix from an archive file

!/#/    {next}  # beginning of the root card

{
        prev = substr($0, index($0, "#"));
        gsub(/\\/, "\n", prev);
        while (getline > 0) {
                sub(/^[ \t]+/, "", $0);
                gsub(/\\/, "\n", $0);
                card = sprintf("%s%s", prev, $0);
                n = index(card, "Version");   # mark the end
                if (n > 0) {   # done
                        printf("%s", substr(card, 1, n-1));
                        break;
                }
                printf("%s", prev);
                prev = $0;
        }
        exit;
}
```

- *Pdbclean.pl*

```
.#! /usr/local/bin/perl
# pdb_clean
# Makes those pesky pdb files better!
# usage: "pdbclean.pl inputfile > outputfile"

foreach (<>) {
    ($first)=(split)[0];

    if (($first eq "COMPND") or ($first eq "CONECT") or
        ($first eq "END") or ($first eq "TER") or
        ($first eq "REMARK")) {
                print;
        } elsif (($first eq "ATOM") || ($first eq "HETATM")) {
                $rec_name="ATOM";
                ($int,$atom_name)=(split)[1,2];

                # Now, substitute nothing for any numbers:
                $atom_name =~ s/[0-9]+//;

                # Now, read in the x,y,z coordinates
                $x=substr($_, 31, 8);
                $y=substr($_, 39, 8);
                $z=substr($_, 47, 8);

                if ($atom_name ne "LP") {
                    write;
                }
        }
}

format STDOUT =
@<<<<< @>>>  @>>              @###.###@###.###@###.###
$rec_name,$int,$atom_name,$x,$y,$z
.
```

- *Sum_angle_strain.pl*

```
#! /usr/local/bin/perl
# Last update: nov/98, Renata Kover
# sum_angle_strain.pl
# evaluates angle_strain of portions of the molecule by adding
# the total angle strain of the individual atoms while
# taking into account the hybridization of the atom. The output is a
# script to be used with pdb files for raswin.
# usage: sum_angle_strain.pl inputfile(.log) (list of atom numbers)

# error checking
if ($ARGV[0] eq "") {
        print "\n Usage: sum_angle_strain.pl inputfile (list of atom numbers)\n";
        print "\n Evaluates angle_strain of portions of the molecule by adding the \n";
    print " total angle strain of the individual atoms while \n";
        print " taking into account the hybridization of the atom. The output is a \n";
        print " script to be used with pdb files for raswin.\n \n";
        exit;
}

# Open the log file for reading
$file = shift;

# Grab all the atoms for which to add up the strain
@sum_atoms = @ARGV;

print "Working on atoms ";
foreach $atom ( @sum_atoms ) {
        print "$atom ";
}
print "\n";

open (IN, "< $file") or die "ERROR: Cannot find $file?\n";

# First, build the atom number-atom type conversion
$get_names = "";
$line_num = 0;
$input_orientations = 0;
while (<IN>) {
        # Should we begin reading in the atom center/number info?
        if( (/Input orientation/) and ($input_orientations == 0) ) {
                $get_names = "true";
        }

        if( $get_names eq "true" ) {
                ++$line_num;
                next if( $line_num < 6 );
                if( /---------/ ) {
                        $get_names = "false";
                        ++$input_orientations;
                        next;
                }
                ( $center_num, $atom_num ) = split;
                $atom_nums[$center_num] = $atom_num;
                # print "center_num=$center_num atom_num=$atom_num\n";
        }
}

# DEBUG PRINT
# foreach $atom_num ( @atom_nums ) {
#       print "atom_num[$count]=$atom_num\n";
#       ++$count;
# }

# Look in a PERL book and learn how to REWIND a file
close( IN );
open (IN, "< $file") or die "Where is $file?\n";

while (<IN>) {
        # Begin looking for the angle info after these words
```

```
        if (/ Optimized Parameters /) { $opt="t"; }
        next unless ( $opt );

        # Angle info looks like this
        # A1    A(2,1,3)              110.0984         -DE/DX =    0.
        next unless (/\! A/);
        # print;

        # get the desired values out of the file
        if (/ \! A\d+\s+A\((\d+)\,(\d+)\,(\d+)\)\s+(\S+)/) {
                ($a, $b, $c, $deg) = ($1, $2, $3, $4, $5);
                $angle = "A($a,$b,$c)";

                # This is a 2D assoc. array
                $angle{$b}{"$a $c"} = $deg;
        }


}
close (IN);

# Grab the atom names
@atoms_mid = sort bynum keys %angle;
# print @atoms, "\n";


foreach $atom_mid ( @atoms_mid ) {
        @atoms_other = keys %{ $angle{$atom_mid} };

        # print "There are ",$#atoms_other+1," angles involving atom $atom_mid\n";

        # If there are 1 or 6 angles then it is sp3 hybridization
        if ( ( $#atoms_other+1 == 6 ) or ( $#atoms_other+1 == 1 ) ) { $hyb{$atom_mid} =
"sp3"; }
        # Otherwise it is sp2 hybridization
        else { $hyb{$atom_mid} = "sp2"; }

        # loop through each angle and print out a report
        $num_angles = $#atoms_other+1;

        foreach $atom_other ( @atoms_other ) {

                # Figure out the explicit names of the other atoms
                ($a, $c) = split " ", $atom_other;

                # sp2 carbon atom
                if ( ($hyb{$atom_mid} eq "sp2") && ($atom_nums[$atom_mid]==6) ) {
                        # print "Calculating the offset for an sp2 carbon\n";
                        &calc_offset( 120 );

                # sp3 carbon atom
                } elsif ( ($hyb{$atom_mid} eq "sp3") && ($atom_nums[$atom_mid]==6) ) {
                        # print "Calculating the offset for an sp3 carbon\n";
                        &calc_offset( 109.6 );

                # sp3 oxygen atom
                } elsif ( ($hyb{$atom_mid} eq "sp3") and ($atom_nums[$atom_mid]==8) ) {
                        # print "Calculating the offset for an sp3 oxygen\n";
                        &calc_offset( 109.6 );

                # sp3 sulfur atom
                } elsif ( ($hyb{$atom_mid} eq "sp3") and ($atom_nums[$atom_mid]==16) ) {
                        # print "Calculating the offset for an sp3 sulfur\n";
                        &calc_offset( 92 );

                } else {
                        print "Whoa there pardner, I dont understand the hybridization of
atom $atom_mid\n";
                        print "atom_num=$atom_nums[$atom_mid]\n";
                        exit;
                }

                $angle = "A($a,$atom_mid,$c)";
                # printf ("%15s %4s %8.4f %6.2f\n",
```

```
                #       $angle, $hyb{$atom_mid}, $angle{$atom_mid}{$atom_other}, $per );


                # Store the total strain for each mid atom
                $strain{$atom_mid} += $offset ;
        }
        # Normalize for the number of angles involved in the strain
        # calculation of each atom
        $strain{$atom_mid} *= 100/$num_angles;

        printf( "The strain at atom_mid, %2d, with hybridization $hyb{$atom_mid}, is %7.3f
percent\n", $atom_mid, $strain{$atom_mid} );

}

# Sum up the total strain
$total_strain = 0;
foreach $atom ( @sum_atoms ) {
        $total_strain += $strain{$atom};
}

# Print a mini strain report
print "The sum of the strain at atoms ";
foreach $atom ( @sum_atoms ) {
        print " $atom";
}
printf( " is %8.4f\n", $total_strain );
exit;

sub calc_offset {
        $ideal = shift;

        # Calculate the normalized deflection from the ideal sp3 angle
        $offset = ( $ideal - $angle{$atom_mid}{$atom_other} ) / $ideal;

        # Square this offset value
        $offset *= $offset;
}

sub bynum { $a <=> $b };
```