Rapid Development of VIPS-NLP: a Visualization Tool for Nonlinear Optimization

Werner E. HELM and Udo SGLAVO, FB MN, Fachhochschule Darmstadt

ABSTRACT and INTRODUCTION

Many problems in business and industry lead to models that require the application of optimization techniques. This applies to all kinds of risk analysis, portfolio analysis, and to many problems in telecommunication development. All linear optimization problems can be solved using the SIMPLEX-algorithm implemented in SAS/OR® Proc LP. Nonlinear optimization however is a much more difficult task. From Rel. 6.10 on SAS/OR® offers the powerful Proc NLP for solution of rather general nonlinear programs. Due to the tricky nature of nonlinear problem-solving strategies the documentation for this single procedure comprises over 320 pages ([2]). There is no graphical user interface available for this procedure. Hence we set out to create one by ourselves using the object-oriented-application development features in SAS/AF® (SAS® for Windows 6.10 and 6.11). We decided to use a rapid prototyping approach. One major goal was the easy visualization of the behaviour of different solution strategies accessible via Proc NLP. This part additionally involved the Procs GCONTOUR, G3D, GANNO, etc. of SAS/GRAPH®.

In this paper we will discuss the key steps in our project starting from setting out the goals, writing our own tutorial (in German) up to using widgets and OLE-controls that just became available with Rel. 6.11. We will summarize our experience and present several examples of the visualization made easily possible via click and point by VIPS-NLP.

VIPS-XXX (Visual Problem Solving using Professional Software) is an expanding series of SAS-based tools for teaching and learning Problem Solving whereby mastering the advanced features of the SAS-System is one of the problems under attack. VIPS-ODE is targeted at the visualization of solution curves of (stiff) systems of ordinary differential equations (ODE) using SAS/IML® Call ODE. VIPS-SIM presently under development will enhance the SAS QSIM-application by macro-based tools for easier replication and evaluation of simulation runs.

GOALS

The project was not devoted to a single goal but aimed at multiple goals. We wanted to make significant progress in each of the following directions:

- provide visualization of NLP-techniques
- create a prototype of a graphical user interface for handling SAS/OR Proc NLP
- use some of the most advanced features of the frame-technology available in SAS/AF to show superiority of the SAS-System over more restricted systems as Mathematica®, etc.
- provide a reference framework for the next future, especially for diploma thesis projects
- provide a tutorial introduction to SAS/AF Frame Entry and enable "Learning by Example" by a thoroughly documented, commented and annotated non-trivial application.

The resulting application VIPS-NLP should be used by professors, students and business analysts for teaching and learning the art of nonlinear optimization and for getting an easy access to NLP-problem solving.

GOALS in DETAIL

As we use Bazaara/Sherali/Shetty [1] as a primary textbook on nonlinear optimization we wanted to be able to reproduce many results used there for demonstration of various aspects of a broad range of optimization algorithms. Especially we wanted to be able to easily reproduce the figures in chapter 8 on pages 303-332. These figures use the function

\[ z = f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2, \]
show contours of that function together with a starting vector $x^0$ and the iterates $x^1, x^2, \ldots$ and show the path on which the chosen solution technique moves towards a (local or global) optimum. Since such plots unavoidably get crowded and barely readable when the algorithm approaches convergence we wanted to zoom about the point of convergence in order to get a clearer/better impression of the behaviour. Having generated good looking graphs we wanted to have at our disposal the full range of possibilities for saving, exporting or editing them.

A frequent weakness in highly interactive systems is the fact that the user can’t keep track of what he really did and where his graphs really belong to. To avoid this trap we thought of a naming-system to save and reference the problems worked on and the graphs being generated. Reasonable defaults should be suggested by the system.

As the SAS/OR-procedures LP and NLP are not accessible via SAS/ASSIST in order to really enable interactive exploration of the power and intricacies of NLP-problem solving we needed to create a GUI for handling Proc NLP.

In an academic setting many sub-projects are tied to assignments and/or parts of diploma thesises. These assignments formally require paper-printed documentation, so a totally undocumented prototype of some kind cannot arise. However since time is always at a premium in these assignments in the past we frequently saw correct software implementations with good paper documentation, but poor GUI (no help, no references to underlying theory, etc). To improve on that difficulty many days have been wasted on transient (i.e. not built to stay) help-compilers of all kind.

Now, MS-Word for Windows (Winword) is by far the most widely used text-processor at our place. Definitely the best way to solve this special help/online documentation problem is to establish dynamic links between the implementation (here the SAS/AF-application VIPS-NLP) and the original paper-documentation. We will show how to accomplish this task with OLE-automation under SAS 6.11 for Windows.

**STARTING POINT**

This is a look at the help-screens of SAS/OR Proc NLP:

**NLP: Syntax**

PROC NLP options;
MIN function names;
MAX function names;
LSQ function names;
MINQUAD matrix, vector, number;
MAXQUAD matrix, vector, number;
PARMS parameter names;
ARRAY array specification;
BOUNDS boundary constraints;
BY variables;
CRPJAC variables;
GRADIENT variables;
HESSIAN variables;
JACOBIAN variables;
LABEL parameter labels;
LINCON linear constraints;
MATRIX matrix specification;
NLINCON nonlinear constraints;
INCLUDE model files;
JACNLCL variables;

**NLP: Procedure Options**

The following options can appear in the PROC NLP statement:

- **Data Set**
  - DATA= SASdataset
  - INEST= SASdataset
  - INQUAD= SASdataset
  - MODEL= model name
  - OUT= SASdataset
  - OUTALL
  - OUTTEST= SASdataset
  - OUTMODEL= model name
  - OUTCRPJAC
  - OUTHESSIAN
  - OUTITER
  - OUTJAC
  - OUTNLJAC
  - NOMISS
  - OUTTIME

- **Optimization**
  - TECHNIQUE= name
  - UPDATE= name

- **Techniques**
  - VERSION= 1|2|3
  - LSPRECISION= r
  - HESCAL= 0|1|2|3
  - INHESSIAN [=r]
  - RESTART= i, i>0
Initial Values,  BEST= i
Grid               INFEASIBLE
RANDOM= i          INITIAL= r
Derivatives       FD [=name|number]
FDHESSIAN [=name]
FDINT= OBJ|CON|ALL
DIAHES
GRADCHECK
OUTDER= 0|1|2
Constraints       LCEPS= r,r > 0
LCDEACT= r
LCSINGULAR= r,r > 0
Termination       MAXFUNC= i          MAXITER= i<n>
Criteria          MINITER= i          MAXTIME= r
ABSCONV= r          ABSFCONV= r<n>
ABSGCONV= r<n>     ABSXCONV= r<n>
FCONV= r<n>        FCONV2= r<n>
GCONV= r<n>        GCONV2= r<n>
XCONV= r<n>        FSIZE= r
XSIZE= r
Covariance        COVARIANCE= 1|2|3|4|5|6 | M|H|J|B|E|U
Matrix            SIGSQ= sq,sq>0
VARDEF= DF|N
ASINGULAR= r,r>0
MSINGULAR= r,r>0
VSINGULAR= r,r>0
COVSING= r,r>0
G4= n,n>0
Printed Output    PALL          NOPRINT
Options           PSHORT        PSUMMARY
PINIT         PHISTORY
PJACOBI       PCRPJAC
PHESSIAN      PNLCJAC
PGRID         PFUNCTION
PSTDERR       PCOV
PEIGVAL       LIST
LISTCODE
Step Length       DAMPSTEP [= r]     MAXSTEP= r<n>
Restriction       INSTEP= r
Miscellaneous     FDIGITS= r         NOEIGNUM
Options           CDIGITS= r         OPTCHECK [=r]
SINGULAR= r,r>0

This is the mathematical problem formulation of Bazaara/Sherali/Shetty [1, p. 309]:

\[ \min \ z = f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2, \quad x = (x_1, x_2) \in \mathbb{R}^2; \]

Start with \[ \mathbf{x}^0 = (0, 3), \] iterate according to Newton-Raphson and draw the first 6 iteration vectors in a \((x_1, x_2)\)-plot; provide additional mathematical information (text output) about the gradient, its norm, the Hessian, inverse Hessian, etc.

The following SAS-code would solve the numerical problem using default values where not specified explicitly. Either NRA or NRR has to be chosen; MAXITER=6 is optional, if omitted the procedure would iterate until convergence.

```
PROC NLP TECH=NRA | NRR | MAXITER=6 | ;
MIN Z ;
PARMS X1= 0, X2= 3 ;
Z=(X1-2)**4+(X1-2*X2)**2 ;
RUN ; QUIT ;
```
However, the graph in [1, p. 311] is not easily generated; neither can the effect of the numerous algorithmic options being described in [1] and/or available in Proc NLP ([2]) be easily demonstrated.

We were well versed in the mathematics of nonlinear optimization, had different levels of SAS-experience, but both were novices to FRAME-programming. We had the documentation available in mid-1995 ([3] to [8]), but the well intentioned tutorial [6] stops way before it really gets interesting. A more extensive and better one was not available then.

ACHIEVING the GOALS

The project was divided in 3 phases. The first one included learning SAS/AF, the basics of SAS® SCL with particular emphasis on the FRAME-entry. This phase resulted in our own tutorial [10, 1st ed., 60 p.] featuring an example of annuity calculations (Rentenberechnung).

Phase 2 was devoted to the development of VIPS-NLP Version 1.0 and to paper-documentation [10, 200 p.]. The first two phases were done with SAS 6.10 for Windows. Phase 3 used SAS® 6.11 for Windows, 1st wave (Orlando®), added bells and whistles, lots of small improvements and OLE-automation for help and online-documentation.

It was clear that no more formal approach could be used: that left rapid prototyping (RAD), where one of the authors took mainly the role of project leader, supervisor and user and the other mainly the role of developer. Some major design decisions were to

- start with functions of (only) two variables
- concentrate at first on minimization problems
- leave out the data-set options
- leave out the data-step options
- put the selection of the algorithmic solution technique on top of VIPS-NLP’s logical menu tree
- try to capture all algorithmic features of Proc NLP via click and point
- add 3-D-visualization through Proc G3D
- implement two graphics modes, the widget „graphics“ (FRAME > Make > Graphics) and the classical SAS/GRAPH gseg - window
- use VIPS-NLP as code generator for larger problems

Leaving out Proc NLP’s data-step- capabilities seemed reasonable since considerable parts of the data-step are available in Proc NLP and trying to create a GUI for the SAS® data-step was deemed beyond our present means.

RESULTS

Figures 1 to 5 show the user-interface of VIPS-NLP. Figures 1 to 5 use a 10 pt. Sasfont (see the discussion below). Since we address users with general interest or even knowledge in the SAS-System we intentionally left the toolbar visible and accessible. In demonstration mode the users have access to a number of well-known functions with completely predefined settings, in interactive mode one can experiment with own choices. The video-interface (cf. figure 2) is up and running, presently playing the SAS®-tiger in the video jungle; in a future release we could play a video featuring Prof. Helm lecturing on Nonlinear Programming. The icons „Theorie“ and „Hilfe“ activate the OLE-automation links to the MS-Winword 6.0-documents.

The icon „Eingabe“ shows the chosen example and a number of options related to the function, the contours and the starting vector. The icon „Einstellung“ shows the control center of Proc NLP’s algorithmic options; those not available for the method chosen (here: NRR) are grayed. As one example the icon/frame „Abbruchkriterien“ is shown in figure 5. The users could get help („Hilfe“) and easily change the preset values. Figure 6 displays the way that the selected and appropriately parameterized Newton-Raphson-Ridge algorithm takes until convergence. If selected appropriately, a lot of additional mathematical information about the gradient, its norm, the Hessian, its determinant and eigenvalue structure etc. can be found in the standard output window which is not shown here.

As was expected the marking of the iterates \( x^j \) gets crowded. One could easily zoom within the range of the iterates \( x^j, j=0,13 \) or look at the surface in 3D; the original graph depends on the usage of colors and can be seen in a frame called „Preview“ using the graphics-widget for display and also in the gseg of SAS/GRAPH. The footnote contains a reference to the graph originating with Beispiel-Funktion 1 for the NRR-method.

Figures 7 to 10 use Beispiel-Funktion 2, Himmelblau’s function [2] which has several local minima to show further aspects of VIPS-NLP and of the convergence behaviour of Proc NLP. Figure 7 shows the small first steps of the NRR-technique before it decides to pursue the lower right local minimum; figure 9 shows zooming...
in on iterate vectors no. 6 to 9; Figure 10 shows the totally different way that the QUASI-NEWTON-method QN takes as compared to NRR: the first large step from $x^0$ to $x^1$ decides that the upper left local minimum is pursued. Using VIPS-NLP’s algorithmic control center many more different choices of mathematical options plus the resulting convergence paths can be easily visualized.

Figure 1
Figure 2
### Figure 4

**Abbruchkriterien**

<table>
<thead>
<tr>
<th>Kriterium</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXFUN (Maximale Anzahl Funktionsaufrufe)</td>
<td>125</td>
</tr>
<tr>
<td>MAXITER (Maximale Anzahl der Iterationen)</td>
<td>50</td>
</tr>
<tr>
<td>MINITER (Mindestanzahl der Iterationen)</td>
<td>0</td>
</tr>
<tr>
<td>MAXTIME (Obergrenze CPU Zeit in Sek.)</td>
<td>1E300</td>
</tr>
</tbody>
</table>

**Absolutes Abbruchkriterium**

<table>
<thead>
<tr>
<th>Kriterium</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSCONV (abs. Funktions-Konvergenz-Krit.)</td>
<td>-1.341E154</td>
</tr>
<tr>
<td>ABSFCONV (abs. Funktions-Konvergenz-Krit.)</td>
<td>0</td>
</tr>
<tr>
<td>ABSGCONV (abs. Gradient-Konvergenz-Krit.)</td>
<td>0.00001</td>
</tr>
<tr>
<td>ABSKCONV (abs. Parameter-Konvergenz-Krit.)</td>
<td>0</td>
</tr>
</tbody>
</table>

**Relative Abbruchkriterien**

<table>
<thead>
<tr>
<th>Kriterium</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCONV (rel. Funktions-Konvergenz-Krit.)</td>
<td>2.2204E-16</td>
</tr>
<tr>
<td>FCONV2 (rel. Funktions-Konvergenz-Krit.)</td>
<td>0</td>
</tr>
<tr>
<td>GCONV (rel. Gradient-Konvergenz-Krit.)</td>
<td>1E-8</td>
</tr>
<tr>
<td>GCONV2 (rel. Gradient-Konvergenz-Krit.)</td>
<td>0</td>
</tr>
<tr>
<td>XCONV (rel. Parameter-Konvergenz-Krit.)</td>
<td>0</td>
</tr>
<tr>
<td>FSIZE (Parameter für FCONV, GCONV)</td>
<td>0</td>
</tr>
<tr>
<td>XSIZE (Parameter für XCONV)</td>
<td>0</td>
</tr>
</tbody>
</table>

### Figure 5

Minimierung mit Iterations-Historie

Funktion: Z=(x1-2)^2+(x1-x2)^2+...
Figure 6

Minimierung mit Iterationen - Historie
Funktion: \( Z = (X_1^2 + X_2 - 11)^2 + (X_1 + X_2^2 - 7)^2 \)

Figure 7

Funktion: \( Z = (X_1^2 + X_2 - 11)^2 + (X_1 + X_2^2 - 7)^2 \)

Figure 8
Figure 9

Minimierung mit Iterationen—Historie
Funktion: \( Z = (x1+x2-1)xx2+(x1+x2xx2-7)xx2 \)

Figure 10

Minimierung mit Iterationen—Historie
Funktion: \( Z = (x1xx2+x2-11)xx2+(x1+X2xx2-7)xx2 \)
Finally we show the code that has been generated by VIPS-NLP via point and click for producing the iterates shown in figure 10. This could easily be edited to solve a problem with, say, 5 variables using the same algorithmic options.

```sas
PROC NLP TECH=QN PALL OUTEST=TEST1
UPDATE=DBFGS LIS=2 INHESS=1
MAXIT=300 MAXFU=800 MINIT=4
MAXTIME=1E300 ABSCONV=-1.341E154 LSP=0.4
ABSGCONV=1E-7 GCONV=1E-10 GCONV2=0
ABSFCONV=0 FCONV=2.2204E-16 FSIZE=0
ABSEXCONV=0 XCONV=0 XSIZE=0
FCONV2=0 SING=1E-8
FDIGITS=15.65356 CDIGITS=15.65356 INSTEP=1
MAXSTEP=1E300 COV=1 SIGSQ=1
VARDEF=DF ASING=1E-150 MSING=1E-12 VSING=1E-8
COVSING=1E-8
G4=60
OUTITER;
MIN Z;
PARMS X1= 0, X2= -3;
;
Z=(X1**2+X2-11)**2+(X1+X2**2-7)**2;
RUN ; QUIT ;
```

IMPLEMENTATION

Much to our surprise the video-connection was the easiest to establish. When editing a FRAME via BUILD just follow the sequence:
RMB > Selection List: OLE-Insert Object > Object Type : Medien-Clip (Create New ; Display as Icon) > OLE-Insert Object Attributes . Entry : VIDEO > Medien-Wiedergabe : Choose name of your avi-file (path\tiger.avi, e.g.) . Confirm with OK, Close or End . Doubleclick your new icon „Medien-Clip“ and you will see and hear the SAS-tiger. You have created a catalogue-entry called „VIDEO.HSERVICE“.

Monitor/Graphics Resolution

The application VIPS-NLP was developed using 800x600 screen resolution. We believed it to be independent of that resolution, i.e. that it would automatically adjust itself to any higher resolution. That was not true. Using 1024x768 screen resolution we experienced a shift of 1-2 cm down and to the right. After a lot of trial and error we learned that
- adjusting the font-size could remedy the problem
- any region sizes are internally converted to a font-size related unit, so are intrinsically dependent on the font.

We used the Sasfont Standard 8 pt. on both resolutions. We would demand that 9-, 11-point and 13-16-point sizes of that font generally be made available by SAS Institute.
Just trying some of the new True Type SAS Monospace fonts in different sizes did not produce good looking results for our existing AF-application.

We advice anyone as early as possible in the development to try different font-types and -sizes, different screen resolutions and at best to keep 2 cm away from the screen border (on all sides).

OLE-Automation

For reasons of consistency, maintainability and convenience we did not not want to split the existing documentation into many separate parts (e.g. as HSERVICES) and we did not want to glue it to the application. Hence OLE-automation was the technique of choice. We simply put text-marks at the jump-to-points in our MS-Word for Windows 6.0 documents „hilfe.doc“ and „theorie.doc“ and stored these two files in the same
directory (libnamed „vips_nlp“) containing our basic SAS-catalogue „nlp.sc2“ . The following segment then implemented the OLE-automation :

INIT:
    length docname $40;
submit continue;
    data one;
      path=iutilpn("vips_nlp");
      if (path NE '') then
        call symput ('msdoc',trim(path)||"\hilfe.doc");
    run;
quit;
endsubmit;
docname=symget('msdoc');
return;
........................................................
HELP:
/*Aufruf Hilfe*/
hostcl = loadclass('sashelp.fsp.hauto');
call send (hostcl, '_NEW_', WordObj, 0, 'Word.Basic');
call send (WordObj, '_DO_', 'DateiÖffnen', docname );
call send (WordObj, '_DO_', 'BearbeitenGeheZu','Markel');
return;

The first part gets the path of the file „hilfe.doc“ and the full file name into the SCL-variable „docname“. The hostcl-command creates a new instance of an OLE-automation object, the method _NEW_ establishes the link to Word for Windows and the messages _DO_ transmit two Word macro commands. The last one jumps to the right place in the document set before as „Markel“.

Most users in Germany use the German version of MS Word for Windows with the default setting German Macro language . We note that the SAS 6.11 for Windows -provided sample program „OLE Automation Example Using Microsoft Word“ will not work in this environment. To get it work we had to modify the program „Memo.scl“ as follows : remove lines 117 and 158 and change in lines 100 resp. 106 „cursor sascombo“ into „call notify(‘sascombo’,_CURSOR_’)“ and „cursor subject“ into „call notify(‘subject’,_CURSOR_’).“

Once we had our own first example up and running it was a snap to get all help-buttons or -icons done. Knowing the right Word macro commands it is straightforward now to activate some more _DO_-messages via call send.

CONCLUSION

The goals could be fully met. Using VIPS-NLP the full power of NLP is at your fingertip : just point and click. This shows the power of the SAS-System over a broad range of functionality (OR, Graphics, Visualization, Multimedia, OO-Application Development, Rapid GUI-Building (RAD)).
The most frequent heard question from our students was „When will we get VIPS-LP or when will Proc LP be covered by SAS/ASSIST ?“.

Some new features (video) are really great and simply work. Others as OLE-automation require better support (sample programs that work, tutorial, GUI-support). „When developing a system using SAS/AF, the programmer is limited only by experience and his/her imagination“, [ 9, p.60 ] , is more true than ever before. The real problem is getting to that experience fast or reducing the amount of experience needed.
The AF-FRAME Entry opens up exciting new possibilities. SAS Institute should consider providing more comprehensive tutorials for developers who start with the FRAME Entry and want/need to „backfill“ knowledge with SAS SCL esp. the connection between SCL and the SAS data-step and the many simple everyday tricks.
REFERENCES


Acknowledgements:

SAS, SAS/OR, SAS/AF, SAS/GRAPH, SAS/IML are registered trademarks of SAS Institute Inc., Cary, NC, USA. Other brand or product names are registered trademarks of their respective companies.

AUTHORS

Prof. Dr. Werner E. HELM
Statistics & OR
FB MN, FH D
Schoefferstr. 3
D-64 295 DARMSTADT

Dipl.-Math. (FH) Udo SGLAVO

Phone: (06151) - 14 63 85; - 16 86 63
FAX: (06151) - 16 89 75

e-mail: helm@fhdatacom2.fhrz.fh-darmstadt.de