EXTENSION OF THE INTRADAY FIBONACCI-DLL

Christoph T. Wildensee

In my thesis for the "Master of Financial Technical Analysis" (thesis title "**INTRADAY-TRADE-OPTIMIZATION VIA FIBONACCI-DLL**") from April 2021, I presented a way to create a DLL with Lazarus, in which all prices to a share are indicated in the course of the day. This curve of prices represents however only the Change Points over the respective day (update every minute is enough, not every 30 seconds as shown in the thesis). From this the Fibonacci auxiliary / support lines are defined from a firmly defined secant formation to the trend. If after the update the new value rises above or falls below the relevant lines, it is probably a trend reversal. This is not a proof, but in high percentage coherence it is an exciting approach.

Now at this point a representation extension is to take place. On the one hand it concerns a visual representation of the changes, expressed in a line between left and right secant value, the further course with the then current price value and the acting Fibonacci lines (procedure **ShowFiboLines**), on the other hand with a simple representation of rise and fall of the course as plus-minus-string (example +++---+++---+-++- ; procedure **PlusMinusString**). Finally, a database connection to a Microsoft Access database should be possible, if it is existing (procedure **DBexists** and **DBAccess**). The control for this is also shown.

1. Secant Display

The secant display of the continuously changing share price is called up when the share price is updated. Provided that the two values on the left and on the right for the secant formation are unequal empty, drawing is called via

```
if ((valueleft <> 0) and (valueright <> 0)) then ShowFiboLines(valueleft, valueright);
```

```
procedure TForm1.ShowFiboLines(v_left: Double; v_right: Double);
var
  diff1: Double;
  diff2: Double;
  diff3: integer;
  factor1: integer;
  high1: integer;
  width1: integer;
  dir1: String;
  right1: Double;
  left1: Double;
  nv: Double;
  posend1: integer;      // y-position of right end point to display new value = right secant point
begin
 right1:=v_right;
 left1:=v_left;
 nv:=right1;
 if (length(LabelNewValue.Caption)>1) then      // after latest update = length of new value <> 0
   begin
    nv:=StrToFloat(LabelNewValue.Caption);      // nv = new value
   end;
```

```
 high1:=PaintBox1.Height;                    // paintbox.height in app = 90 points
 width1:=PaintBox1.Width;                    // paintbox.width in app = 462 points
 high1:=round(high1/2)+5;                    // = 90/2 = 45+5 = 50
 width1:=round(width1 -(width1 mod 100)-20);
 dir1:='    ';                               // string includes direction string at Paintbox1
 with PaintBox1 do begin
  Canvas.Clear;                              // paintbox is empty
  diff1:=abs(right1-left1);
  if (diff1 >= 0) then factor1:=round(high1/2);     // factor checks difference of target values to
  if (diff1 >= 1) then factor1:=round(high1/5);     // different modi like a zoom
  if (diff1 >= 2) then factor1:=round(high1/9);     // paintbox: left up: 0,0, right down: 462,90
  if (diff1 >= 3) then factor1:=round(high1/12);    // so need to spread diff-results
  if (diff1 >= 4) then factor1:=round(high1/15);    // that the diffs have more space
// pos. direction
  if (fiboricht1 = true) then
   begin
    dir1:='pos.';
    if (diff1 > 0) then
     begin
    // Trend = right - left
      diff3:=round(diff1*factor1);
      Canvas.Pen.Color := clRed;
      Canvas.Line(0, high1, width1, high1-diff3);
      posend1:=high1-diff3;
    // Fibonacci Auxiliary lines
    // now f162
      diff2:=abs(right1-altef162a);
      diff3:=round(diff2*factor1);
      Canvas.Pen.Color := clGreen;
      Canvas.Line(width1-50, high1-diff3, width1, high1-diff3);
    // now f150
      diff2:=abs(right1-altef150a);
      diff3:=round(diff2*factor1);
      Canvas.Pen.Color := clBlue;
      Canvas.Line(width1-50, high1-diff3, width1, high1-diff3);
    // now f138
      diff2:=abs(right1-altef127a);
      diff3:=round(diff2*factor1);
      Canvas.Pen.Color := clYellow;
      Canvas.Line(width1-50, high1-diff3, width1, high1-diff3);
    // now die Linien andere Seite
    // now f38
      diff2:=abs(right1-altef38a);
      diff3:=round(diff2*factor1);
      Canvas.Pen.Color := clYellow;
      Canvas.Line(width1-50, high1-diff3, width1, high1-diff3);
    // now f50
      diff2:=abs(right1-altef50a);
      diff3:=round(diff2*factor1);
      Canvas.Pen.Color := clBlue;
      Canvas.Line(width1-50, high1-diff3, width1, high1-diff3);
```

```
    // now f62
      diff2:=abs(right1-altef62a);
      diff3:=round(diff2*factor1);
      Canvas.Pen.Color := clGreen;
      Canvas.Line(width1-50, high1-diff3, width1, high1-diff3);
    // Textout important information like secant left/right & direction & factor
    Canvas.TextOut(0,0,'F-Step: '+EditFStepLeft.Text+'_'+EditFStepRight.Text+'   Direction:
'+dir1+'   New Value: '+FloatToStrF(nv, fffixed, 5,2)+'    Factor: '+IntToStr(factor1));
     end;
   end;
  // neg. direction
    if (fiboricht1 = false) then
     begin
      dir1:='neg.';
      if (diff1 > 0) then
       begin
      // Trend = right - left
        diff3:=round(diff1*factor1);
        Canvas.Pen.Color := clRed;
        Canvas.Line(0, high1, width1, high1+diff3);
        posend1:=high1+diff3;
      // now f162
        diff2:=abs(right1-altef162a);
        diff3:=round(diff2*factor1);
        Canvas.Pen.Color := clGreen;
        Canvas.Line(width1-50, high1+diff3, width1, high1+diff3);
      // now f150
        diff2:=abs(right1-altef150a);
        diff3:=round(diff2*factor1);
        Canvas.Pen.Color := clBlue;
        Canvas.Line(width1-50, high1+diff3, width1, high1+diff3);
      // now f138
        diff2:=abs(right1-altef127a);
        diff3:=round(diff2*factor1);
        Canvas.Pen.Color := clYellow;
        Canvas.Line(width1-50, high1+diff3, width1, high1+diff3);
      // now die Linien andere Seite
      // now f38
        diff2:=abs(right1-altef38a);
        diff3:=round(diff2*factor1);
        Canvas.Pen.Color := clYellow;
        Canvas.Line(width1-50, high1+diff3, width1, high1+diff3);
      // now f50
        diff2:=abs(right1-altef50a);
        diff3:=round(diff2*factor1);
        Canvas.Pen.Color := clBlue;
        Canvas.Line(width1-50, high1+diff3, width1, high1+diff3);
      // now f62
        diff2:=abs(right1-altef62a);
        diff3:=round(diff2*factor1);
        Canvas.Pen.Color := clGreen;
```

```
      Canvas.Line(width1-50, high1+diff3, width1, high1+diff3);
    // values as a text on paintbox.canvas
      Canvas.TextOut(0,0,'F-Step: '+EditFStepLeft.Text+'_'+EditFStepRight.Text+'   Direction:
'+dir1+'   New Value: '+FloatToStrF(nv, fffixed, 5,2)+'    Factor: '+IntToStr(factor1));
     end;
   end;
  PaintBox1.Canvas.TextOut(0,14,'l: '+FloatToStrF(uelinks, fffixed, 5,2));
  PaintBox1.Canvas.TextOut(width1+8, 14,'r: '+FloatToStrF(uerechts, fffixed, 5,2));
 // last part of the line in red like trend line
 diff2:=right1-nv;                                  // block ok, because without abs()
 if (fiboricht1 = true) then                        // pos. direction
  begin
    diff3:=round(diff2*factor1);
    Canvas.Pen.Color := clRed;
    Canvas.Line(width1, posend1, width1+30, posend1+diff3);
  end;
  if (fiboricht1 = false) then                      // neg. direction
   begin
    diff3:=round(diff2*factor1);
    Canvas.Pen.Color := clRed;
    Canvas.Line(width1, posend1, width1+30, posend1+diff3);
   end;
 end;
 PlusMinusString(StringGridValues, 4, 'x', 3, LabelPlusMinus);   // start Plus-Minus-String
end;
```

It is nessecary to show the target of that data-driven implementation. The Points A and B as the left start and right end point of the secant develop in the right direction to A' and B' every time when a new change point is identified.

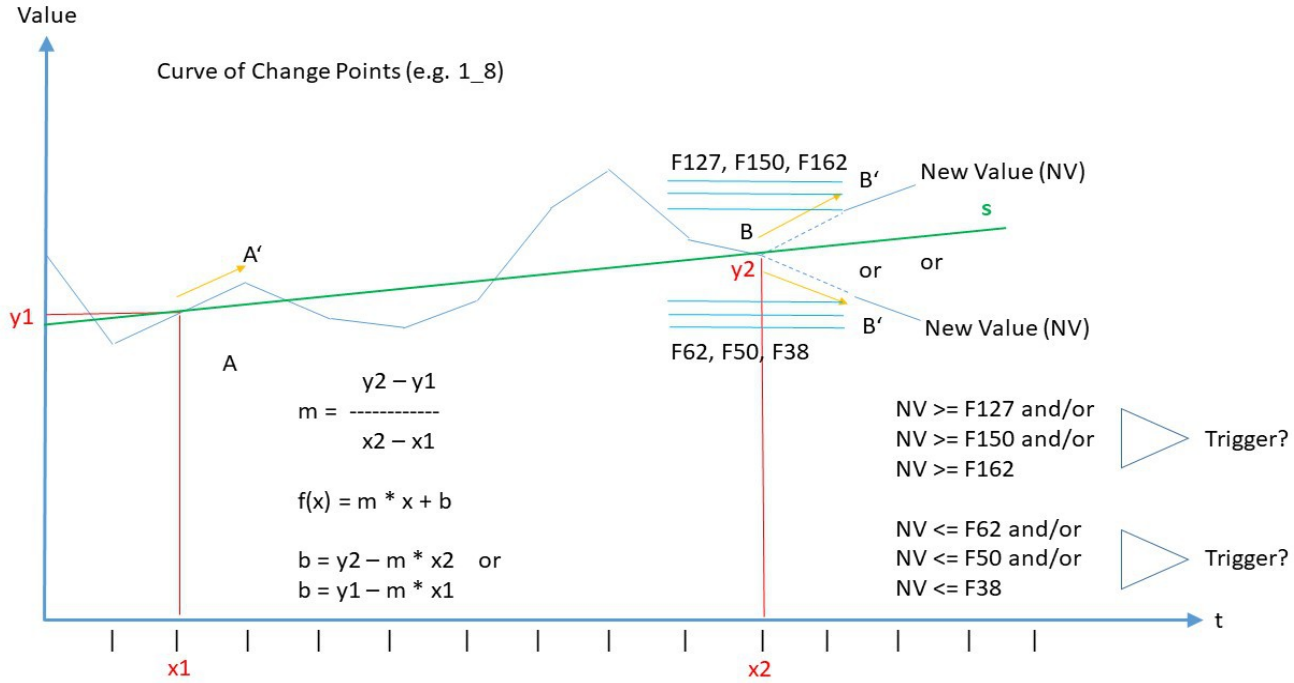

Figure 1: Showing the mathematical base / Trend Reversal Trigger.

At this point it is necessary, too, to show a curve with negative direction. So we can see that the support lines are defined above and below starting from the right point of the secant as the master point with 100%, the above lines with 62, 50, 38%, the lines below as a result with 127, 150, 162%. It is only interesting, that the blocks of 38-62 and 127-162 are correctly defined.

Figure 2: Negative Direction of Fibonacci Support lines [METATRADER5].

2. Recognition of the Curve Progression

It is useful to be able to recognize with a quick glance how the values have changed during the day. This is of course already done by the TChart - element. Nevertheless, I would like to show how the last 20 values of the course of the price can be displayed via the variable "plusminus1" as a plus-minus-string.

```
procedure TForm1.PlusMinusString(SG: TStringGrid; ColSeperator: integer; Seperator1:
String; ValueColPosition: integer; LabelPlusMinus: TLabel);
[...]
plusminus1:='';
for i:=1 to maxvalues do
 begin
  Value1[i]:='';                  // Array of Double to store values of the stock (every minute)
 end;
for i:=1 to 20 do
 begin
  Diff1[i]:=0;                    // we need only the last 20 differences (max)
 end;
LabelPlusMinus.Visible:=false;                    // Label to show the Plus-Minus-String
LabelPlusMinus.Caption:='';
LabelPlusMinus.Refresh;
```

```
// Ident and Insert Value
for i:=SG.RowCount-1 downto 1 do
 begin
  if ((TRIM(SG.Cells[ColSeperator,i]) = Seperator1)) then  // Seperator for change points is „x“
  begin
   anz:=anz+1;
   Value1[anz]:=SG.Cells[ValueColPosition, i];        // store values min. 7, max. 20
  end;
 end;
 p:=0;
 plusminus1:='';
 if (anz > 6) then                                    // min. 7 => if not 7, don't want to display
  begin
   q:=(anz-19);
   if (q <= 1) then q:=2;
// create delta and show
   for i:=anz downto q do
    begin
     p:=p+1;
     try
      valuest1:=StrToFloat(Value1[p]);
      valueen1:=StrToFloat(Value1[p+1]);
      diffvalues:=(valuest1-valueen1)*-1;
      Diff1[p]:=diffvalues;
     except
       //
     end;
    end;
// plusminus1 and display in Label LabelPlusMinus.Caption
   for i:=1 to 20 do
    begin
     if (Diff1[i] < 0) then plusminus1:=plusminus1+'+';        //vice versa, because diffvalues * -1
     if (Diff1[i] > 0) then plusminus1:=plusminus1+'-';        // equal 0 = not relevant
    end;
    end;
   i:=length(plusminus1);
// plusminus1 vice versa
   if (i >= 7) then        // only if length(plusminus1) >=7
    begin
     for t:=i downto 2 do
      begin
       k:=k+copy(plusminus1, t-1, 1);
      end;
     plusminus1:=k;                                     // showing the invers string
     LabelPlusMinus.Visible:=true;
     LabelPlusMinus.Caption:=plusminus1;
     LabelPlusMinus.Refresh;
    end;
```

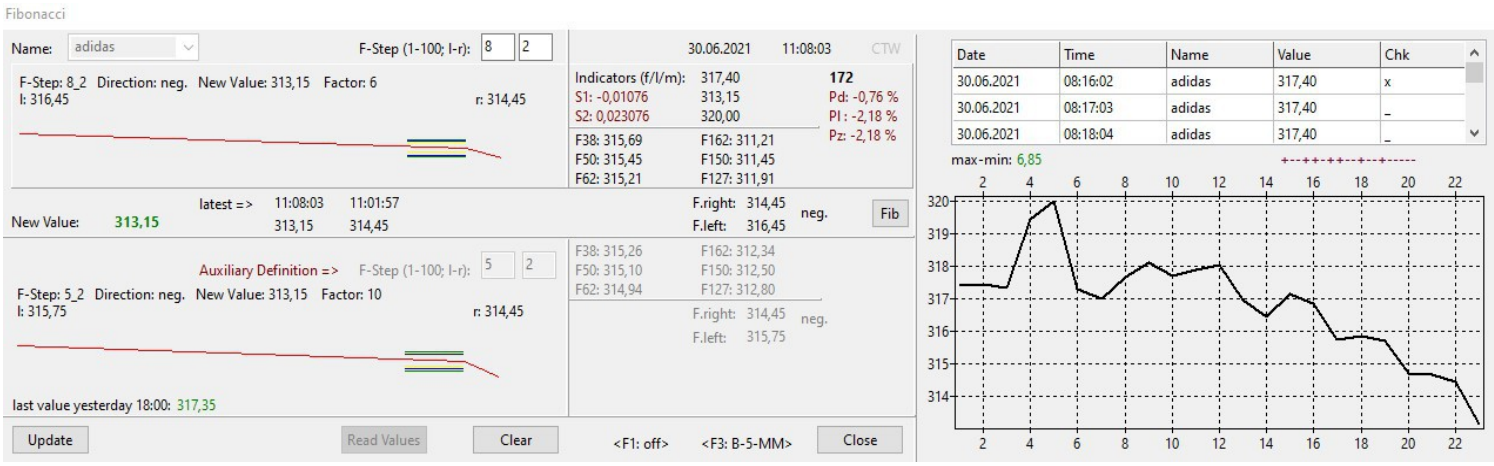

Figure 3: Application with Extension (Fibonacci Support Lines, left + right points of the secants).

Figure 4: Plus-Minus-String.

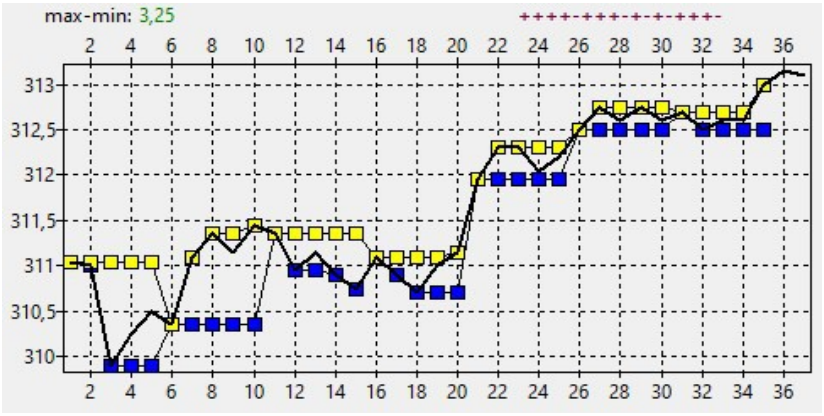

Figure 5: F3-Extension (5-Block, Max-Min-Points).

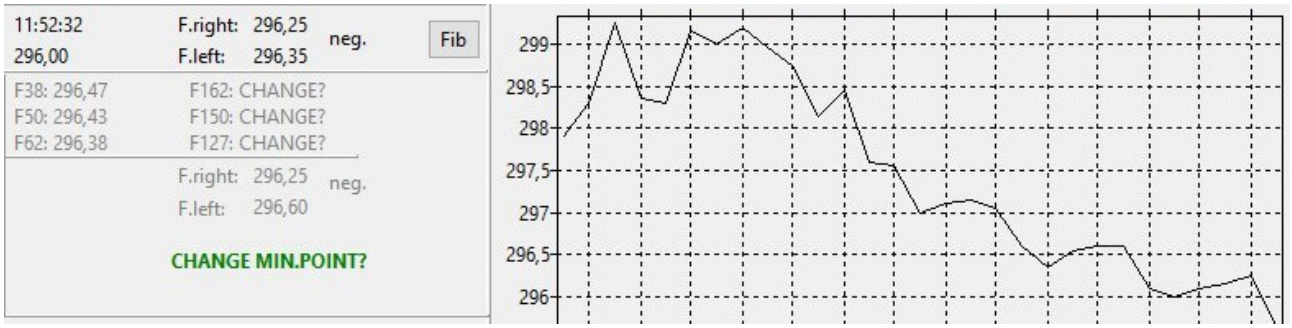

Figure 6: Maybe change at the point of latest minimum value.

3. Database Access

The programmed functionality asks on exit / quit if the data should be saved as txt file on c:\temp. It may be useful to save the data additionally in a (local) database, here Microsoft Access, to be able to evaluate / analyze them later. So if **in the same directory as the DLL or Application** the file **HistLine.dsn** with the content

```
[ODBC]
DRIVER=Microsoft Access Driver (*.mdb)
UID=admin
UserCommitSync=Yes
Threads=3
SafeTransactions=0
PageTimeout=5
MaxScanRows=8
MaxBufferSize=2048
FIL=MS Access
DriverId=25
DefaultDir=
DBQ=db_name.mdb
```

and the **target database** (here db_name.mdb) are available, the data is stored there. The appropriate structure setup in the database is assumed (table Hist and the corresponding fields as shown in the SQL-INSERT-Statement). The call to the database check is made via

```
 if (MessageDlg('Quit:', 'Do you want to quit?', mtConfirmation,
  [mbYes, mbNo],0) = mrYes)
  then
   begin
    DBexists;                          // dbexists1 = application variable = false or true
    if ((dbexists1 = true) and (StringGridValues.RowCount-1 > 1)) then
     begin
          if (MessageDlg('Save in Database:', 'Do you want to save the Values in the Database?',
mtConfirmation,
      [mbYes, mbNo],0) = mrYes)
      then
       begin
        DBAccess(datnam1);             // store with name in variable datnam1
       end;
     end;
    Application.Terminate;
   end;
```

First, it is checked whether the database exists at all.

```
procedure TForm1.DBexists;
begin
 dbexists1:=true;
 //check connection
 try
```

```
  ODBCConnection1.Connected:=false;
  ODBCConnection1.FileDSN := ExtractFilePath(ParamStr(0))+'HistLine.dsn';
  ODBCConnection1.Connected := true;
  ODBCConnection1.KeepConnection := true;
  SQLTransaction1.DataBase := ODBCConnection1;
  SQLTransaction1.Action := caCommit;
  SQLTransaction1.Active := true;
  SQLQuery1.DataBase := ODBCConnection1;
  SQLQuery1.UsePrimaryKeyAsKey := false;
  SQLQuery1.SQL.Text := 'select * from Hist';        // Hist is the table in the database
  DataSource1.DataSet := SQLQuery1;
  SQLQuery1.Open;
 except
  dbexists1:=false;                                  // if not correct, then dbexists1=false
 end;
 try
  ODBCConnection1.Connected := false;
 except
  //
 end;
end;
```

Saving the data in the database starts with the following procedure.

```
procedure TForm1.DBAccess(Titel: String);
var
 title1: String;
 i1:integer;
begin
// *
 i1:=0; title1:=Titel;
// Store in Database (must be the same path as the application / dll; correct db-structure is essential)
[...]
// when DB exists and Values are in the StringGrid
   if ((dbexists1 = true) and (StringGridValues.RowCount-1 > 1)) then // values in StringGrid
    begin
    try
     SQLQuery1.SQL.Clear;
     SQLQuery1.SQL.Add
('INSERT INTO Hist
(Content, Date1, Time1, Name1, Value1, Chk, F_left, F_right, F38, F50, F62, F127, F150, F162,
F38_Change, F50_Change, F62_Change, F127_Change, F150_Change, F162_Change,
MinMaxChange) '+
 'VALUES (:a, :b, :c, :d, :e, :f, :g, :h, :j, :k, :l, :m, :n, :o, :p, :q, :r, :s, :t, :u, :v);');
     for i1:=1 to StringGridValues.RowCount-2 do
      begin        // transfer variables to the database, stringgrid.cols store data
       SQLQuery1.ParamByName('a').Value := title1;
       SQLQuery1.ParamByName('b').Value := StringGridValues.Cells[0,i1];
       SQLQuery1.ParamByName('c').Value := StringGridValues.Cells[1,i1];
       SQLQuery1.ParamByName('d').Value := StringGridValues.Cells[2,i1];
       SQLQuery1.ParamByName('e').Value := StringGridValues.Cells[3,i1];
```

```
      SQLQuery1.ParamByName('f').Value := StringGridValues.Cells[4,i1];
      SQLQuery1.ParamByName('g').Value := StringGridValues.Cells[5,i1];
      SQLQuery1.ParamByName('h').Value := StringGridValues.Cells[6,i1];
      SQLQuery1.ParamByName('j').Value := StringGridValues.Cells[7,i1];
      SQLQuery1.ParamByName('k').Value := StringGridValues.Cells[8,i1];
      SQLQuery1.ParamByName('l').Value := StringGridValues.Cells[9,i1];
      SQLQuery1.ParamByName('m').Value := StringGridValues.Cells[10,i1];
      SQLQuery1.ParamByName('n').Value := StringGridValues.Cells[11,i1];
      SQLQuery1.ParamByName('o').Value := StringGridValues.Cells[12,i1];
      SQLQuery1.ParamByName('p').Value := StringGridValues.Cells[13,i1];
      SQLQuery1.ParamByName('q').Value := StringGridValues.Cells[14,i1];
      SQLQuery1.ParamByName('r').Value := StringGridValues.Cells[15,i1];
      SQLQuery1.ParamByName('s').Value := StringGridValues.Cells[16,i1];
      SQLQuery1.ParamByName('t').Value := StringGridValues.Cells[17,i1];
      SQLQuery1.ParamByName('u').Value := StringGridValues.Cells[18,i1];
      SQLQuery1.ParamByName('v').Value := StringGridValues.Cells[19,i1]; // 20 Cols (0-19)
      SQLQuery1.ExecSQL;
      SQLTransaction1.Commit;
     end;
    except
     ShowMessage('Store into Database was not successful!');
    end;
[...]
```

At the end, it is important, too, to display differences between values, in this example the difference between the max and min value of the trading day. Additionally the last value on yesterday can be displayed, because there you can see the starting trend for today. See Figure 3.

```
procedure TForm1.ButtonUpdateClick(Sender: TObject);
[...]
// LabelEndYesterday => search last value yesterday
SuchStart := 0;
SuchStart := FindInMemo(MemoHTMLText, 'Vortag', SuchStart + 1); // search string in side code
if (SuchStart > 0) then
LabelTextPosition.Caption := 'Found at Memo1.SelStart['+IntToStr(SuchStart)+'] !'
// if found, store position
else
LabelTextPosition.Caption := 'No position found!'; // else 0
LabelTextPosition.Refresh; // show String with value
LabelEndYesterday.Font.Color:=clGreen;
LabelEndYesterday.Caption:=copy(MemoHTMLText.Text, SuchStart+35 , 15);
if (length(LabelEndYesterday.Caption) = 0) then LabelEndYesterday.Caption:='./.';
Wertzerlegen1(LabelEndYesterday.Caption); // String cut, only value, no chars
try
diffyester:=(StrToFloat(LabelH.Caption)-StrToFloat(LabelEndYesterday.Caption));
// LabelH = highest value of the day = max
LabelEndYesterday.Caption:=LabelEndYesterday.Caption+' / diff. (value first today - last
yesterday): '+FloatToStrF(diffyester, fffixed, 5,2);
except
//
end;
```

```
  if (diffyester < 0) then LabelEndYesterday.Font.Color:=clMaroon;
  LabelEndYesterday.Refresh;
end;
```

(and in the Part of RateCourse at the end: // *procedure TForm1.KursBewerten*;)

```
[...]
  if ((length(low1)>0) and (length(max1)>0)) then
   begin
[...]
    diffmaxminusmin:=StrToDouble(max1)-StrToDouble(low1);
   end;
 if (diffmaxminusmin > 0) then
  begin
   try
    LabelMaxMinusMin.Caption:=FloatToStrF(diffmaxminusmin, fffixed, 5,2);
    LabelMaxMinusMin.Font.Color:=clGreen;
    LabelMaxMinusMin.Refresh;
   except
     //
   end;
  end;
end;
```

4. Conclusion

4.1 Important Settings

With this functionality it is possible to see the course of the day change points, but it is clear that it does not work in the first hour of the trading day, because there are not enough values. The program works well only at the moment, when **both secants are filled** (if not, TLabel in red with caption: DO NOT PAY ATTENTION. NOT ENOUGH VALUES !). After that hour it works accurately, unless the curve development makes too many movements in too many short intervals. This leads to unspecific auxiliary / support lines, that offer no help while the change points are displayed. But you can see every time the trend of the curve development. In a final version, perhaps the buttons „Read Values“ and „Clear“ should be invisible after all.

I decided to change the F-Step definition. In my thesis, I showed the F-Step like 2_8 (means two steps from right point, eight change points in the left direction). The better way as shown in the application is to change it into 8_2, because the data input is shown in this direction. However, this is not really essential. (Notice: The best starting time of the DLL is near 8:00 in the morning (german time zone), so that it starts already at 9:00 with some past change points and can develop time-optimized support lines between 9:00 and 10:00, that near 10:00 the support lines are showing according to the expected / anticipated results. Otherwise there is a need to wait maybe more than one hour to get correct interpretation.)

Additionally, there is a need to be sensitive with the IT-System Formats, because if the IT-System is not in a german format setting, the getting of values and date and time formation are points of crashing the application. So, the DefaultFormatSettings must be implemented like

```
DefaultFormatSettings.DecimalSeparator := ',';
DefaultFormatSettings.ThousandSeparator := '.';
DefaultFormatSettings.DateSeparator:= '.';
DefaultFormatSettings.TimeSeparator:= ':';
DefaultFormatSettings.CurrencyDecimals:=2;
DefaultFormatSettings.ShortDateFormat:= 'dd.mm.yyyy';
DefaultFormatSettings.ShortTimeFormat:= 'hh:nn';
DefaultFormatSettings.LongTimeFormat:= 'hh:nn:ss';
```

in the FormShow Event.

4.2 Theoretical supplements

In principle, models that persuade the user that a share price trend is only to be considered as a function of time and thus suggest that time is the only relevant factor and that other influencing variables can be completely relegated to the background, are problematic. Time certainly plays an important role, but there are also considerable influences and correlations with other variables such as press releases, annual financial statements, competitive situation and cooperations, even geostrategic aspects (see e.g. Gazprom) etc., which do not result in clearly definable variable assignments [see e.g. Schmelzer (2009), p. 21ff., 125]. In these "stochastic processes, the essential goal of time series analysis is to fit an existing time series into a linear model in such a way that the structure of the data is reproduced as best as possible, and the estimated model parameters allow a forecast in the form of a statistical extrapolation." [Schmelzer (2009), p. 28; translated] It should not be hidden that such models have their demonstrable successes and that research supports them under certain conditions [see e.g. Rostan et. al. (2020)]. To this extent, such a digression is not included here. Maybe it will be more a topic with AI focus. But it should be noted that for the above time series analyses in (higher versions of) Delphi, among other things, there is a paid supplement available via „Dew Lab“ / „Dew Research“ [DEWLAB]. (Machine Learning Extensions can maybe be implemented with Python4Delphi, it should work also with Lazarus.) In addition, several pages should be included that show the development of stock prices, even time-shifted. It is to be left open whether an assistance via Fibonacci support lines is more suitable. However, the presented procedure should also not show the suitability, but only the possibility of the uncomplicated functional integration.

4.3 Way to Extensions

Finally, the approach shows that different users get identical results when looking at the same data. So the approach is comprehensible and reproducible. It can be assumed that it can be further developed and, if necessary, enriched with additional data.
The handling of Lazarus (Delphi [Embarcadero] equivalent; cross platform Rapid Application Development) and its outstanding possibilities of the problem / solution structuring and programming is to be emphasized. It is unproblematic with a little programming experience to develop high-quality applications, this I had already explained in 2020 [see Wildensee (2020)]. Also the production of DLLs opens a further field for the functional extension of (foreign) software products under Microsoft Windows [have additionally a look to DLL via Matlab & Delphi: Sun/Cui/Xu (2014)]. See this complete project at Wildensee (2021a) and Wildensee (2021b).

July, 2021

References

DEWLAB: Dew Lab Studio for Delphi 2021, High performance numerical, statistical and dsp library for Delphi/C++, https://dew-lab-studio-for-delphi.software.informer.com/2021/ .

METATRADER5: MetaTrader 5 Help - Fibonacci Retracement, https://www.metatrader5.com/en/terminal/help/objects/fibo/fibo_retracement .

Rostan et. al. (2020): Options trading strategy based on ARIMA forecasting, emerald 2020, https://www.emerald.com/insight/content/doi/10.1108/PRR-07-2019-0023/full/pdf?title=options-trading-strategy-based-on-arima-forecasting .

Schmelzer (2009): Die Volatilität von Finanzmarktdaten - Theoretische Grundlagen und empirische Analysen von stündlichen Renditezeitreihen und Risikomaßen, Diss. Köln 2009, https://kups.ub.uni-koeln.de/2730/1/dissertation.pdf .

Sun/Cui/Xu (2014): Design and Implementation of a Time-frequency Analysis System for Non-stationary Vibration Signals Using Mixed Programming, International Journal of Hybrid Information Technology Vol. 7, No. 6 (2014), pp. 283-294, https://gvpress.com/journals/IJHIT/vol7_no6/24.pdf .

Wildensee (2020): Fachbereichsspezifischer Entwicklungsansatz zur bedarfsgerechten Analyse SQL-basierter Datenquellen. In: PRev Revisionspraxis 1/2020, S. 7-13, http://www.wildensee.de/laz1.pdf .

Wildensee (2021a): http://www.wildensee.de/fibo_ext.zip .

Wildensee (2021b): http://www.wildensee.de/AlsDLLAufruf.zip .