Jan Ruchel, Łukasz Świontek

Zastosowanie Visual Basic do tworzenia pomocniczych aplikacji dla geodetów

Acta Scientifica Academiae Ostroviensis nr 32, 95-105

2009

Jan Ruchel
Łukasz Świontek

ZASTOSOWANIE VISUAL BASIC DO TWORZENIA POMOCNICZYCH APLIKACJI DLA GEODETÓW

Pomiary wykonywane w geodezji związane z pomiarem sieci, zarówno poziomych jak i pionowych, dostarczają dużej ilości danych które następnie trzeba opracować. Opracowanie danych z tych pomiarów polega na wyrównaniu wyznaczanych wartości oraz określeniu dokładności z jaką został wykonany pomiar, a także określenie błędów jakimi są obarczone wyniki wyrównania.

Realizacja obliczeń w pracach geodezyjnych wymaga posiadania przez geodetę odpowiedniej aplikacji, umożliwiającej ich wykonanie. Pracownicy firm mają więc do wyboru zakupienie programu wykonującego interesujące ich obliczenia lub też mogą spróbować stworzyć własna aplikację do ich realizacji, z możliwością ustalenia szczegółów prezentacji wyników.

Własny program może realizować zagadnienia, od najprostszych obliczeń po skomplikowane analizy. To my zadecydujemy jakie funkcje będzie realizowała napisana przez nas aplikacja.

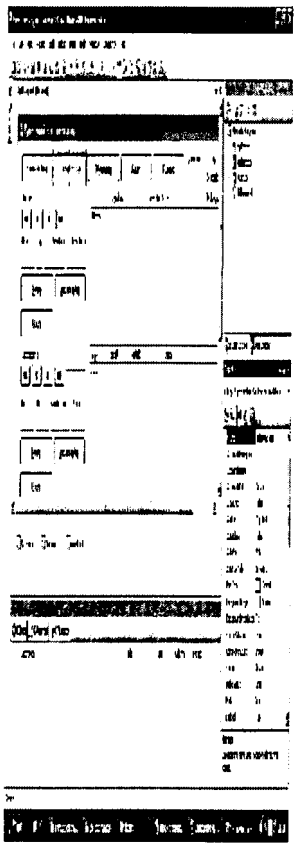
Komercyjne programy dostępne obecnie na rynku oferują gotowe wyniki, nie pozwalając na wgląd do wyników pośrednich, będących niejednokrotnie niezbędnymi danymi do wykonania analizy, szczególnie przy „kłopotach" z obliczeniami. Pisząc własny program do obliczeń możemy zapewnić sobie wgląd do pośrednich wyników w dowolnym interesującym nas etapie obliczeń.

Aplikację wykonującą interesujące nas obliczenia można wykonać na wiele sposobów:

- napisać program od podstaw, wykorzystując istniejące na rynku narzędzia do programowania (np. Visual Basic, Pascal, C++, Delphi),
- wykorzystać popularny program Excel, wchodzący w skład Microsoft Office. Posiada on wbudowane elementy programowania makropoleceń, przy użyciu języka Visual Basic, co również umożliwia tworzenie kompletnej aplikacji.

W niniejszej pracy pokazany zostanie przykład wykorzystania języka programowania Visual Basic do stworzenia aplikacji wyrównującej sieci niwelacyjne.

Visual Basic wywodzi się z bardzo popularnego języka BASIC, wykorzystywanego do nauki programowania. Programiści zaczynający „swoją przygodę" z programowaniem, używali go zanim przeszli na bardziej zaawansowane języki programowania. Twórcy tego narzędzia programistycznego postawili sobie za cel stworzyć najprostszą drogę dla wszystkich, którzy chcą programować dla własnych, jak i komercyjnych potrzeb (co też w znacznym stopniu się powiodło). Pomimo swojej prostej budowy stanowi on bardzo atrakcyjne narzędzie do tworzenia własnych aplikacji.



Program posiada bardzo przyjazny interfejs umożliwiający wygodne tworzenie aplikacji. Po prawej stronie okna głównego programu znajdują się;

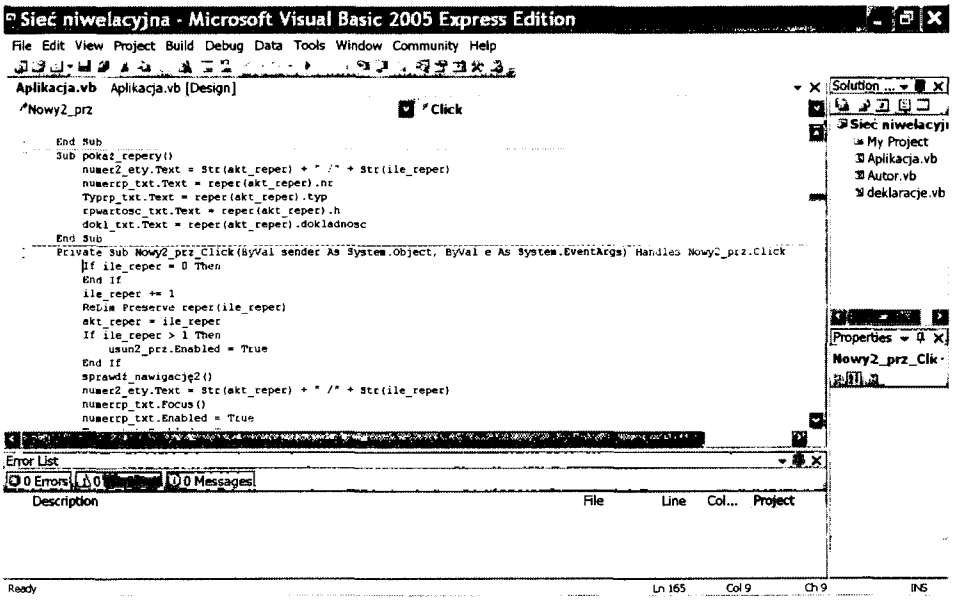
- okno projektu (Solution Explorer) zawierające wszystkie części składowe naszego projektu Są to odpowiednie katalogi i pliki wygenerowane automatycznie przez Visual Basic, oraz formy i moduły dodane w trakcie tworzenia projektu przez programistę,
- okno cech (Properties) pokazujące aktualne wartości wszystkich cech dla aktywnego w oknie projektu obiektu. W oknie tym możemy modyfikować wartości wybranych cech dla wskazywanego obiektu. W razie potrzeby cechy mogą być zmieniane z poziomu kodu programu, podczas realizacji poszczególnych procedur.

Rysunek 1 Okno projektu i okno cech obiektu.

Program tworzony za pomocą Visual Basic składa się z dwu części:

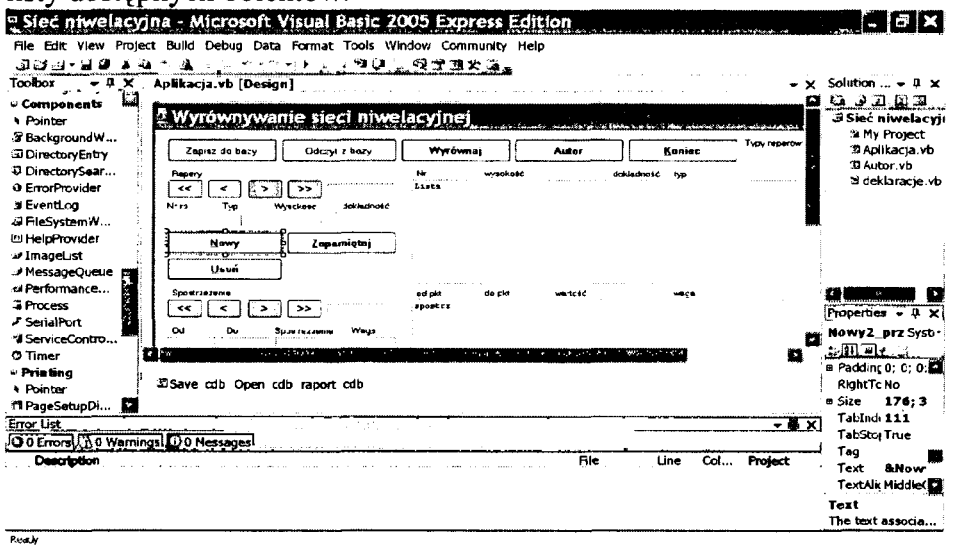
- interfejs użytkownika, czyli graficzny układ okna (formularza) i obiektów do niego przypisanych, Oczywiście projekt rozbudowany może składać się z wielu formularzy (okien).
- kod programu, zawierający odpowiednie procedury z zestawem instrukcji, które maja być wykonane po zaistnieniu odpowiedniego zdarzenia, powiązanego z obiektem (tworzony program jest typu „zdarzeniowego" – czyli wystąpienie, lub nie wystąpienie odpowiedniego zdarzenia warunkuje kolejność wykonywanego kodu programu).

W środkowej części okna głównego Visual Basic (w oknie projektowania – View Designer) w trybie graficznym można projektować wygląd naszej aplikacji. W następnym etapie tworzy się kod programu W oknie kodu programu – View Code), składający się z procedur przypisanych do poszczególnych obiektów i powiązanych z nim zdarzeń.



Rysunek 2 Okno Visual Basic z fragmentem kodu aplikacji do wyrównania sieci niwelacyjnej w trybie edycji kodu programu.

W trybie graficznym programu po lewej stronie umieszczone jest okno (Toolbox) przybornika, zawierające (odpowiednio pogrupowane) listy dostępnych obiektów.



Rysunek 3 Okno Visual Basic z fragmentem projektu (interfejsu użytkownika) aplikacji do wyrównania sieci niwelacyjnej.

Dodawanie obiektów do naszego projektu polega na umieszczeniu ich za pomocą kursora myszki na formularzu i określenie jego wielkości i położenia przy wykorzystaniu myszy i mechanizmów dotyczących elementów graficznych w środowisku Windows. Wykorzystując okno właściwości (Properties) ustalamy wartości poszczególnych cech dla danego obiektu.

Aplikacje napisane w tym języku są programami typu zdarzeniowego, oznacza to, że odpowiednia procedura do realizacji jest wywoływana po zaistnieniu odpowiedniego zdarzenia powiązanego z wybranym obiektem. Zdarzenie, czyli np. wykonanie przez użytkownika czynności takiej jak: kliknięcie klawiszem myszy, wciśnięcie odpowiedniego klawisza na klawiaturze, zaznaczenie kursorem itp., bądź też zdarzenie wynikające z upływu czasu. Zdarzenie musi być powiązane z konkretnym obiektem.

Procedura obsługi zdarzenia na następującą składnię:

```
Private Sub Przycisk_click (parametry
procedury)
....
'lista instrukcji przypisanych do obiektu
„PRZYCISK" i zdarzenia „CLICK"
.....
end sub
```

Jak widać, nazwa procedury zawiera nazwę obiektu i skojarzone z tym obiektem zdarzenie. Ewentualnie zawiera odpowiednie parametry powiązane z rodzajem obiektu i typem zdarzenia. Nagłówki procedur są generowane automatycznie przez Visual Basic.

Visual Basic jest przyjaznym językiem programowania ze względu na wbudowaną kontrolę błędów. Błędy pojawiające się podczas pisanie programu można podzielić na kilka grup:

- pomyłki w edycji tekstu – można je ominąć poprzez staranne skontrolowanie tego, co napisaliśmy, nie są to błędy groźne dla powstałej aplikacji,
- błędy ujawniane podczas kompilacji – są to błędy które uniemożliwiają skompilowanie programu, ponieważ w kodzie programu są nieścisłości (niezgodności składni dla deklaracji bądź instrukcji), lub błędy identyfikacji poszczególnych elementów programu.
- błędy wykonania programu – błędy te pojawiają się podczas wykonywania fragmentu algorytmu, może to np. być przypadek, gdy wystąpi dzielenie przez 0. Wystąpienie takiego błędu jest groźne w skutkach, ponieważ powoduje ono zawieszenie programu, a tym samym stratę danych (tych które nie zostały zapisane). Należy się więc stosować odpowiednie zabezpieczenia podczas pisania kodu programu.
- błędy logiczne – jest to najgroźniejszy rodzaj błędów, gdyż powoduje on uzyskiwanie błędnych wyników. Jest to błąd najtrudniejszy do wykrycia, ponieważ nie jest on sygnalizowany żadnymi komunikatami. Jest to najczęściej wynik błędnego algorytmu.

Program Visual Basic posiada wbudowane mechanizmy pozwalające na zabezpieczanie się i kontrolę błędów wykonywania programu.

Przystępując do pisania aplikacji musimy mieć starannie opracowane założenia dotyczące celu, jaki stawiany jest przed tworzonym programem oraz opracowany i sprawdzony algorytm postępowania, prowadzący do zrealizowania postawionego celu.

Tworząc aplikację do obliczeń musimy zaprojektować (i zarezerwować na te dane odpowiednie zasoby pamięci) odpowiednie struktury, gdzie będą przechowywane nasze dane. Działanie to nazywamy deklarowaniem. Nasze stałe bądź też zmienne są najczęściej typu:

- typ liczby całkowite – Integer
- typ liczby rzeczywiste – Single
- typ logiczny – Boolean
- typ ciąg znaków – String
- typ macierzowy – Array

Visual Basic pozwala także na tworzenie własnych typów złożonych, czyli takich, które będą odpowiadać danym geodezyjnym. Przykładem mogą być tablice zawierające odpowiednio dane typu:

- reper – reprezentowany przez numer, wysokość, klasę, dokładność wyznaczenia wysokości
- punkt osnowy – reprezentowany przez numer, współrzędne, dokładność wyznaczenia, klasę itp..

Mając już tak zadeklarowane zmienne, oraz wartości stałe występujące w naszej aplikacji, możemy przystąpić do pisania właściwego kodu programu.

Pisząc kod programu niejednokrotnie zdarza się sytuacja wymagająca, aby pewien fragment kodu wykonywał się wielokrotnie, bądź też będzie nałożony warunek, w którym będzie determinowane wykonanie określonego kodu.

Do rozwiązania takiego problemu możemy się posłużyć pętlami. Do najpopularniejszej pętli należy blok instrukcji For.

Ma ona następującą budowę:

```
For i=1 to n
        'blok instrukcji powtarzanych
```

```
Next i
```

Powyższa instrukcja pozwala nam na wykonanie „n" – razy ciągu instrukcji, dla określonego (zmieniającego się) parametru „i".

Bardzo ważną konstrukcją, także często występującą w kodzie programu jest blok instrukcji warunkowej:

```
If (warunek) then
 'blok instrukcji dla spełnionego warunku
Else
 'blok instrukcji dla niespełnionego warunku
End If
```

Dla powyższego przykładu instrukcji warunkowej przypisane zostały dwa bloki instrukcji zależne od warunku. W przypadku, gdy zostanie spełniony warunek, zostanie wykonany pierwszy blok instrukcji, zaś drugi zostanie pominięty.

Znając podstawy programowania każdy może napisać taki program komputerowy, który umożliwi wykonanie dowolnego procesu obliczeń geodezyjnych. Do wykonania takiej aplikacji konieczna jest wiedza potrzebna do rozwiązania problemu.

Możemy rozróżnić następujące typowe sieci geodezyjne

- Sieć niwelacyjna,
- Sieć liniowa,
- Sieć kątowa,
- Sieć kątowo-liniowa i
- Sieć przestrzenna.

Do rozwiązania poszczególnych zadań konieczna jest znajomość algorytmu prowadzącego do rozwiązania zagadnienia.

W przypadku obliczania sieci niwelacyjnych należy postępować według następującego algorytmu (algorytm wyrównania metodą najmniejszych kwadratów) :

1. Obliczenie ilości spostrzeżeń i niewiadomych, oraz ustalenie wag

Warunkiem koniecznym do rozwiązania zadania jest posiadanie większej liczby spostrzeżeń niż niewiadomych, czyli:

$$n_k = n - u$$

gdzie: n – liczba spostrzeżeń (pomierzonych ciągów niwelacyjnych)
u – liczba niewiadomych (wyznaczanych reperów)
n_k – liczba spostrzeżeń nadliczbowych

Wagi spostrzeżeń są konieczne do zrównoważenia spostrzeżeń o różnej dokładności. Dla sieci jednakowo dokładnej przyjmuje się, że wagi są sobie równe (z wartością =1), dla różno dokładnych wagi możemy wyznaczyć:

$$p_i = \frac{\sigma_0^2}{\sigma_0^2 L_i} = \frac{1}{L_i}$$

gdzie: L_i – długość i-tego ciągu niwelacyjnego
lub

$$p_i = \frac{1}{n_i}$$

gdzie: n_i – liczba stanowisk w danym ciągu niwelacyjnym

2. Obliczenie wartości przybliżonych
Obliczamy je na podstawie znanej wysokości, co najmniej jednego punktu, do której dodajemy kolejne różnice (lub sumy różnic) wysokości między reperami mierzonymi (zgodnie z wykonanym pomiarem).

$$H_A^P = H_P + \sum h_i$$

3. Ułożenie równań poprawek

$$(H_A^P + dH_A) - H_P = h_1^{obs} + v_1$$
$$(H_A^P + dH_A) - (H_B^P + dH_A) = h_2^{obs} + v_2$$

Równania poprawek w zapisie macierzowym:

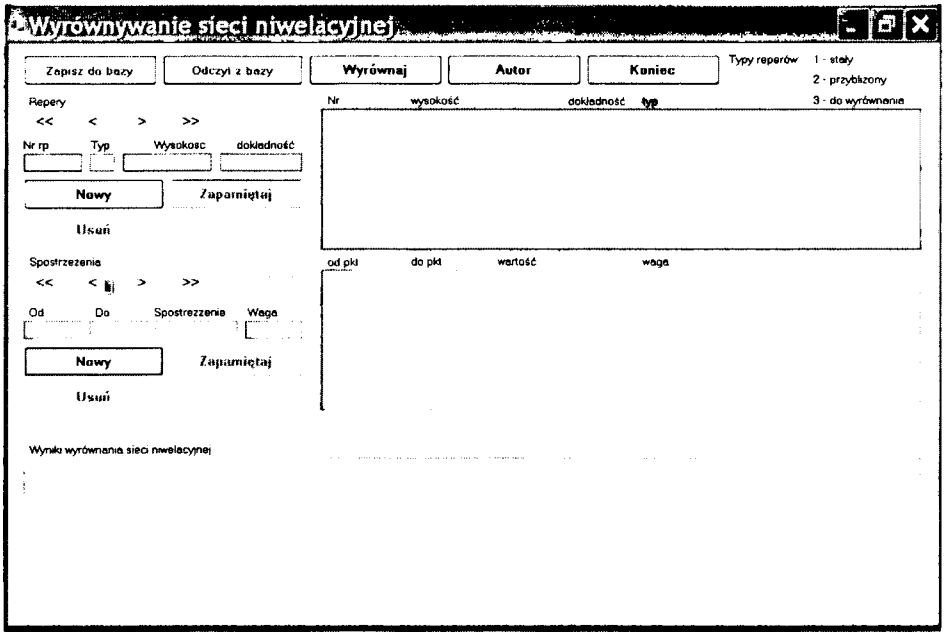
$$V = Ax + L$$

gdzie: **V**- macierz wartości poprawek
A – macierz współczynników przy niewiadomych w układzie równań poprawek
x – macierz niewiadomych
L – macierz wyrazów wolnych w równaniach poprawek

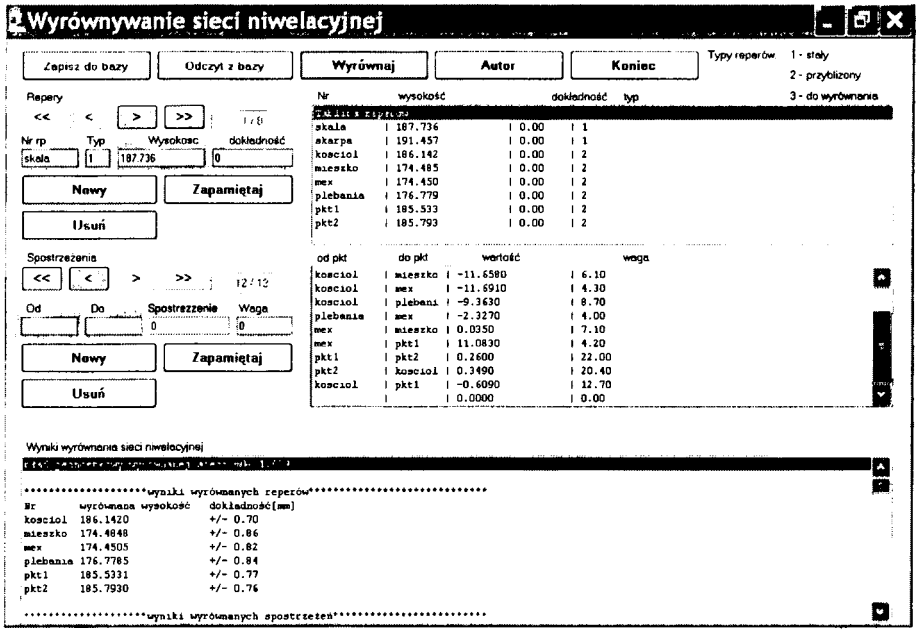
4. Rozwiązanie układu równań normalnych

$$(A^T pA)^{-1} * x = (A^T pL)$$
$$x = (A^T pA)^{-1} * (A^T pL)$$

5. Obliczenie współrzędnych wyrównanych
$$H_A = H_A^P + dH_A$$
6. Obliczenie poprawek do spostrzeżeń
Do obliczenia poprawek do spostrzeżeń i zarazem wyrównanych ich wartości korzystamy z równań poprawek w postaci macierzowej.
$$V = Ax + L$$
7. Wykonanie analizy dokładności.
Rozpoczynamy od błędu średniego typowego spostrzeżenia (estymatora wariancji resztowej).
$$m_0 = \pm\sqrt{\frac{[pvv]}{n-u}}$$
Aby obliczyć dokładność wyznaczonych przyrostów do przybliżonych wysokości reperów należy obliczyć macierz cov(x), której elementy przekątniowe są kwadratami błędów na poszczególnych punktach.
$$\mathrm{cov}(x) = m_0^2 \cdot (A^T pA)^{-1}$$
Podobnie jak w przypadku obliczenia dokładności wyrównanych spostrzeżeń (funkcji niewiadomych)
$$\mathrm{cov}(L) = A \cdot \mathrm{cov}(x) \cdot A^T$$
Macierze te można wykorzystać do wykonania poszerzonej analizy dokładności.
8. Zestawienie wyników
Piszemy kod programu tak, aby interesujące nas wyniki były prezentowane na ekranie monitora w żądanej postaci i ewentualnie udostępniamy możliwość zapisania ich w pliku (np. tekstowym).
Przykładowe okna ze zrealizowanej ww. sposób aplikacji



Rysunek 4 Okno startowe gotowej aplikacji do wyrównywania sieci niwelacyjnych.



Rysunek 5 Okno główne (po wykonaniu obliczeń) gotowej aplikacji do wyrównywania sieci niwelacyjnych.

Literatura:

1. Instrukcja techniczna O-1, Warszawa 1988
2. Kamela C., Warchałowski E., Włoczewski F., Wyrzykowski T.: Niwelacja Precyzyjna, Niwelacja geometryczna, trygonometryczna, satelitarna i hydroniwelacja, PPWK, Warszawa-Wrocław 1993.
3. Łoś A.: Rachunek wyrównawczy, PWN, Warszawa 1973.
4. MacDonald M.: Visual Basic 2004 Wprowadzenie do programowania
5. Osada E.: Geodezja, Oficyna Wydawnictwa Politechniki Wrocławskiej, Wrocław 2001.
6. Świontek Ł.: Program do wyrównywania sieci niwelacyjnej (Visual Basic), praca dyplomowa WSBIP WGiK Ostrowiec Świętokrzyski, rok 2008.