SETTINGSDLG()

Quick reference:

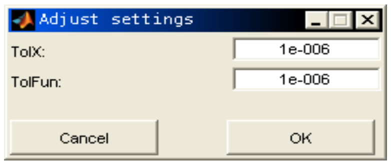
settings = settingsdlg('fieldname', default_value, ...)
settings = settingsdlg({'display string', 'fieldname'}, default_value, ...)
settings = settingsdlg(..., 'title', 'window_title', 'description', 'short_description')
settings = settingsdlg(..., 'separator', 'sep_string', ...)
settings = settingsdlg(..., 'checkbox_string', true, ...)
settings = settingsdlg(..., 'checkbox_string', [true true], ...)

Manual:

The function settingsdlg() is a GUI-dialog much like MATLAB's default errordlg(), questiondlg() and warndlg(), which provides a standardized way to assign specific values to a structure. This structure can then be used to insert specific settings into one of MATLAB's many standard algorithms, or your own. The most basic usage is as follows:

```
settings = settingsdlg(...
    'TolX'   , 1e-6,...
    'TolFun', 1e-6);
```

which will produce the following dialog:



After pressing "OK" or "Cancel", the structure `settings` will be

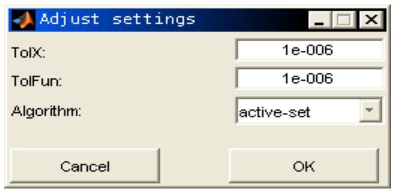
```
settings =

      TolX: 1.0000e-006
    TolFun: 1.0000e-006
```

or any value you assigned to these fields. Naturally, you can add as many fields as you want; the dialog will automatically adjust its size to match your input. If you would like the user to not just insert numeric values, but select values from a string list, use

```
settings = settingsdlg(...
    'TolX'      , 1e-6,...
    'TolFun'    , 1e-6,...
    'Algorithm', {'active-set','interior-point'});
```

which will produce the following dialog:



The resulting structure then is

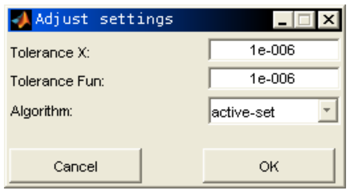
```
settings =

         TolX: 1.0000e-006
       TolFun: 1.0000e-006
    Algorithm: 'active-set'
```

Of course, it isn't always convenient to have the text for each option in the dialog box equal to the fieldname in the resulting structure. If you want the fieldname to be different from the displayed string, you can use something like:

```
settings = settingsdlg(...
    {'Tolerance X'  ;'TolX'  }, 1e-6,...
    {'Tolerance Fun';'TolFun'}, 1e-6,...
    'Algorithm', {'active-set','interior-point'});
```

which produces the box



```
Adjust settings
Tolerance X:        1e-006
Tolerance Fun:      1e-006
Algorithm:          active-set
```

and associated structure

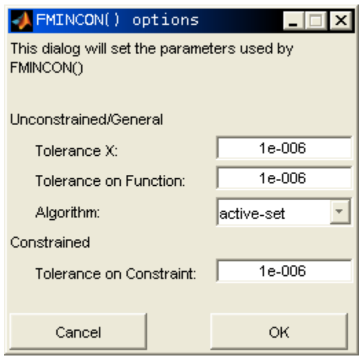
```
settings =

         TolX: 1.0000e-006
       TolFun: 1.0000e-006
    Algorithm: 'active-set'
```

Also, you can add separators, a different dialog title, and a brief description:

```
settings = settingsdlg(...
    'Description', 'This dialog will set the parameters used by FMINCON()',...
    'title'      , 'FMINCON() options',...
    'separator'  , 'Unconstrained/General',...
    {'Tolerance X'  ;'TolX'  }, 1e-6,...
    {'Tolerance on Function';'TolFun'}, 1e-6,...
    'Algorithm'  , {'active-set','interior-point'},...
    'separator'  , 'Constrained',...
    {'Tolerance on Constraint';'TolCon'}, 1e-6);
```

which produces



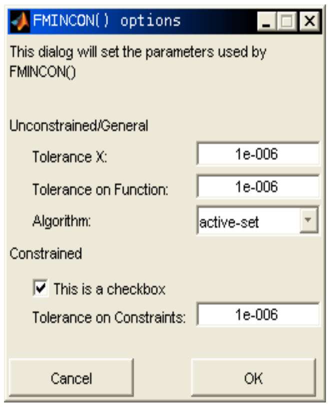
```
FMINCON() options
This dialog will set the parameters used by
FMINCON()

Unconstrained/General
    Tolerance X:                1e-006
    Tolerance on Function:      1e-006
    Algorithm:                  active-set
Constrained
    Tolerance on Constraint:    1e-006
```

The 'title' and 'description' options can appear anywhere in the argument list, they will not affect the fields in the output structure. The order of the 'separator' option of course *does* matter, but, it will *not* be added as a field to the output structure. You can also use logicals, which produce checkboxes:

```
settings = settingsdlg(...
    'Description', 'This dialog will set the parameters used by FMINCON()',...
    'title'      , 'FMINCON() options',...
    'separator'  , 'Unconstrained/General',...
    {'Tolerance X';'TolX'}, 1e-6,...
    {'Tolerance on Function';'TolFun'}, 1e-6,...
    'Algorithm'  , {'active-set','interior-point'},...
    'separator'  , 'Constrained',...
    {'This is a checkbox'; 'Check'}, true,...
    {'Tolerance on Constraints';'TolCon'}, 1e-6);
```

which produces


FMINCON() options

This dialog will set the parameters used by FMINCON()

Unconstrained/General

Tolerance X: 1e-006

Tolerance on Function: 1e-006

Algorithm: active-set

Constrained

☑ This is a checkbox

Tolerance on Constraints: 1e-006

and

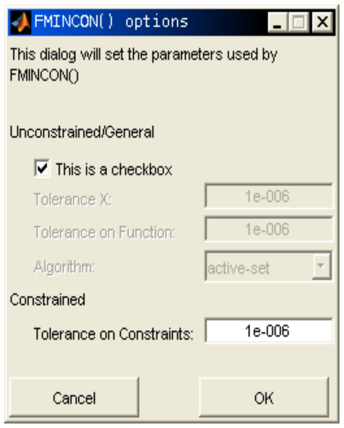
```
settings =

         TolX: 1.0000e-006
       TolFun: 1.0000e-006
    Algorithm: 'active-set'
        Check: 1
       TolCon: 1.0000e-006
```

You can also assign multiple (logical!) values to a single checkbox, in which case the fields below the checkbox are all disabled/enabled when you check it:

```
settings = settingsdlg(...
    'Description', 'This dialog will set the parameters used by FMINCON()',...
    'title'      , 'FMINCON() options',...
    'separator'  , 'Unconstrained/General',...
    {'This is a checkbox'; 'Check'}, [true, true],...
    {'Tolerance X';'TolX'}, 1e-6,...
    {'Tolerance on Function';'TolFun'}, 1e-6,...
    'Algorithm'  , {'active-set','interior-point'},...
    'separator'  , 'Constrained',...
    {'Tolerance on Constraints';'TolCon'}, 1e-6);
```

Which results in


FMINCON() options

This dialog will set the parameters used by FMINCON()

Unconstrained/General

☑ This is a checkbox

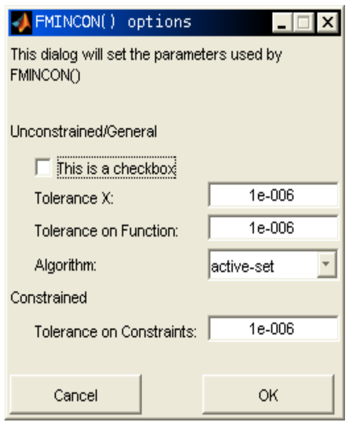
Tolerance X: 1e-006

Tolerance on Function: 1e-006

Algorithm: active-set

Constrained

Tolerance on Constraints: 1e-006


FMINCON() options

This dialog will set the parameters used by FMINCON()

Unconstrained/General

☐ This is a checkbox

Tolerance X: 1e-006

Tolerance on Function: 1e-006

Algorithm: active-set

Constrained

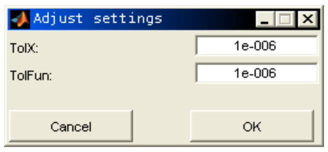
Tolerance on Constraints: 1e-006

Setting the checkbox value to `[true, true]` will cause the dialog box to appear with all fields below the appropriate separator *disabled,* whereas a value of `[true, false]` will have all fields initially *enabled.* Checking or un-checking the checkbox will simply swap the enabled/disabled states.

Finally, you can insert a single structure as a (single!) argument, which produces a dialog box according to its settings and fieldnames:

```
settings = struct(...
    'TolX'   , 1e-6,...
    'TolFun', 1e-6);
settings = settingsdlg(settings);
```

producing


Adjust settings

TolX: 1e-006

TolFun: 1e-006

Naturally, since all other information is absent in this last example, the functionality in this case is rather limited. But if the intent is to change a fairly simple structure, it certainly suffices.

==RECENT CHANGES==

- As requested by Peter Zemp-Hurni, you can now also set the width of the window, as well as the width of each uicontrol:

  ```
  settings = settingsdlg(...
        'Description' , ['This dialog will set the parameters ',...
                         'used by FMINCON()'],...
        'title'        , 'FMINCON() options',...
        'WindowWidth' , 250,...
        'ControlWidth', 100);
  ```

  These values should be given in pixels. Make sure though that `WindowWidth` is larger than `ControlWidth`, otherwise the output will look distorted.
- As suggested by peter, `settingsdlg` now has a second output argument:

  ```
  [settings, button] = settingsdlg(...);
  ```

The `Button` output argument will be equal to `'OK'` when the OK-button is pressed, `'Cancel'` when the Cancel- button is pressed, and empty ([ ]) when the window is closed.

- The separators are now printed boldface and span the whole width of the window (instead of `'ControlWidth'`)

Although this is the third revision, there may still be some bugs left.
Please report any bugs you find to oldnhuis@gmail.com.