

Power Flow for the HP Prime

`Flujo de Potencia.hpprgm` — a single program with a graphical interface that solves power flows on the calculator, with five methods:

Method	Type	What it reports
Gauss-Seidel	iterative	complex voltages, the P / Q used to update V , calculated powers, losses
Newton-Raphson	iterative	$H \ N \ M \ L$, reduced Jacobian, mismatches, corrections, losses
Decoupled	iterative	exact H and L per iteration, corrections, losses
Fast decoupled	iterative	constant B' and B'' , corrections, losses
DC power flow	direct	B' matrix, angles, injections and line flows

It is the same application as `Flujo de Potencia.tns` for the TI-Nspire, rewritten in PPL with its own interface: title bar, editable tables, a scrollable result viewer and touch support.

On a TI-Nspire? The same program, with the same five methods, is available for that calculator at github.com/SCLDarkKnight/Newton-Raphson-Gauss-Seidel-TI

The on-screen text is in Spanish. Every label you will see on the calculator is quoted here exactly as it appears, with the English meaning next to it the first time it shows up.

Este documento también está disponible en [español](#).

Made with love by **Bernhardt Gotschlich** · linkedin.com/in/bernhardt-gotschlich

1. Installing

Requirements: an HP Prime (G1 or G2), physical or the desktop emulator, with recent firmware. It does not need the calculator's Python interpreter: everything is written in PPL.

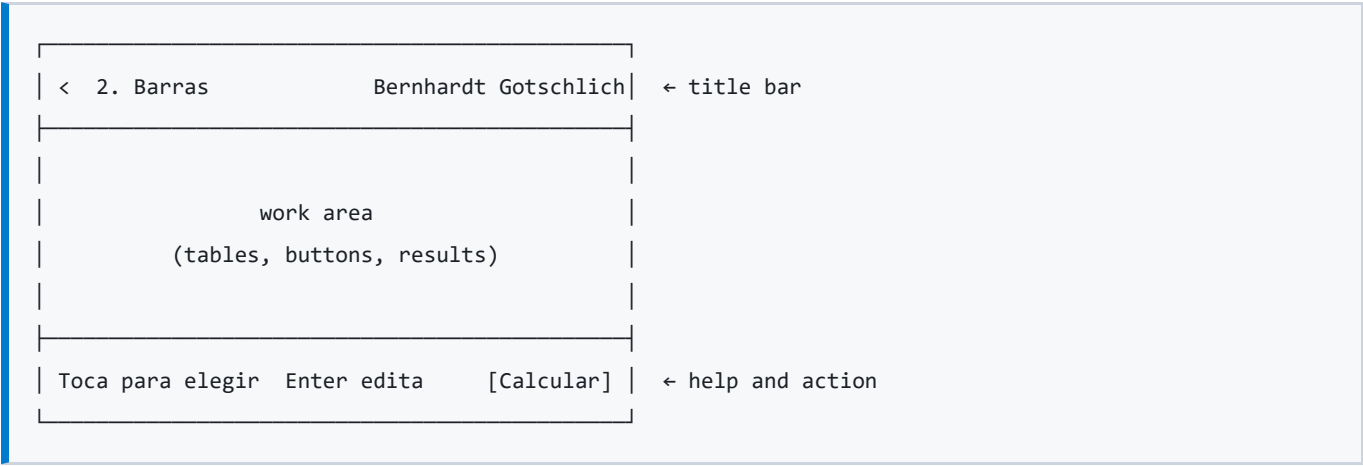
1. Connect the calculator and open the **HP Connectivity Kit**.
2. Drag `Flujo de Potencia.hpprgm` onto the **Programs** node of your calculator in the left-hand panel.
3. On the calculator, open the **Program Catalog** with **Shift + 1 (Program)**, pick `Flujo de Potencia` and press **Run**.

You can also launch it from the home screen by typing `FLUJO()` and **Enter**, or from **Toolbox > User > Flujo de Potencia > FLUJO**.

To quit, press **Esc** from the Home screen. If the calculator freezes, hold **On** down for a few seconds.

2. The screen

The HP Prime screen is 320×240 . The program lays it out like this:



The blue **<** button at the top left always returns to the previous screen, even in the middle of editing a cell. It appears on every screen except Home.

The blue **Calcular** (*Calculate*) button at the bottom right appears on the data screens — Sistema, Barras, Líneas and Matriz Y — and runs the chosen method.

3. Controls

Key	Action
arrows	move the cursor through the table / scroll the results
Enter	edit the cell, press the button, change the bus type
<i>typing a digit</i>	starts editing the cell right away
Esc	cancel the edit, go back, or quit from Home
⌫ (backspace)	while editing, deletes the last digit
(-) (change sign)	types the minus sign
Menu	calculates with the chosen method, from any data or result screen
+	on the Líneas screen, adds a row
-	on the Líneas screen, deletes the row under the cursor

The screen is **touch-sensitive** everywhere: tap buttons, cells, the **<** button and the **Calcular** button. On the results screen, tap the upper half to scroll up seven lines and the lower half to scroll down seven.

Tapping a table cell **only moves the cursor**. To change a value, press **Enter** or start typing the number.

On the **Líneas** screen the **-** key deletes the row. If you need to type a negative value there — a series compensation reactance, say — use the **(-)** change-sign key, not the keyboard minus.

4. Workflow

Home → Sistema → Barras → Líneas or Matriz Y → Calcular → Results

- 1. **Home** (Inicio) — pick the method, or load the 3-bus example.
 - 2. **Sistema** (System) — bus count, how you will enter the network, method and iterations.
 - 3. **Barras** (Buses) — the type of each bus and its electrical data.
 - 4. **Líneas** (Lines) or **Matriz Y** (Y matrix) — the network topology.
 - 5. **Calcular** (Calculate) — jumps to the results.
- You can move back and forth with **<** or **Esc** without losing anything: the data stays put as long as the program is open. Switching methods **does not clear** the data, so you can solve the same system with all five methods and compare.

5. Conventions (read this before entering data)

Item	Convention
Units	everything in per unit (pu) on a common base . The program does not handle MVA or kV
Sign of P and Q	it is injected power: generation positive , load negative . A 50 MW load on a 100 MVA base is entered as P = -0.5
Angles	always in radians (the DC flow also shows degrees)
Shunt Bsh	enter the total for the line; the program puts half at each end (π model)
Y matrix	symmetric: you enter only the upper triangle and the rest fills itself in
Slack bus	there must be exactly one . Marking a bus as SL turns the previous slack into PQ automatically
Decimals	4 are displayed; the arithmetic runs at the calculator's full precision

6. Bus data

The **Barras** screen. One row per bus, seven columns:

Column	What it is
#	bus number (not editable)
Tipo	type: SL (slack), PV or PQ — Enter cycles through them
P	scheduled active power (pu)
Q	scheduled reactive power (pu)
V esp	voltage the bus is regulated to (pu) — SL and PV only
V0	initial voltage estimate (pu) — PQ only
ang0	initial angle (rad)

Cells that **do not apply** to the bus type show a - and cannot be edited. What each type asks for:

Type	You enter	The program computes
SL (slack)	V esp , ang0	P and Q
PV	P , V esp , ang0	Q and the angle
PQ	P , Q , v0 , ang0	V and the angle

The two voltage columns do not overlap, because they cover different cases:

- On **slack and PV** buses the voltage magnitude is **given**: |V|esp fixes it and it does not move during the solution. v0 does not apply there and shows as -.
- On **PQ** buses the magnitude is the **unknown**: v0 is only the starting point. Wherever you start from, it converges to the same answer.

All five methods read |V|esp the same way, so the same system gives the same result with any of them.

7. Entering the network

On the **Sistema** screen you choose between the two modes with ◀ ▶.

Líneas mode (recommended)

The **Líneas** screen. One row per line:

Column	What it is
De / A	from / to: the buses it joins (1..n)
Dato	data: Z if you will give impedance, Y if admittance — Enter toggles
R/G	R if the data is Z ; G if it is Y
X/B	X if the data is Z ; B if it is Y
Bsh	total shunt susceptance of the line (0 if it has none)

Rows are added with + and deleted with -. Each new line starts out as 1 → 2 , Z , R = 0.02 , X = 0.06 , Bsh = 0 .

- If you define **two lines between the same pair of buses**, they merge on their own: the series admittances and the Bsh values are added.
- A line with Z = 0 , or with both ends on the same bus, is ignored.
- This is the only mode that can report **losses per line**.

Matriz Y mode

The **Matriz Y** screen. One row per element of the **upper triangle** (i ≤ j), with columns i , j , G and B . Elements below the diagonal are mirrored automatically, so they do not appear in the table.

In this mode there are **no per-line losses and no DC line flows**: without line data there is no way to compute them. The program says so in the results. The DC flow itself still works: it builds its B' from the imaginary part of the off-diagonal Y .

8. Which method to use

Method	Typical iterations	Good at	Watch out for
Gauss-Seidel	5–15	simple, robust from bad starting points	slow on large networks
Newton-Raphson	2–4	converges very fast, gives the full Jacobian	more sensitive to the start
Decoupled	3–6	half the system to solve, nearly as fast	same
Fast decoupled	4–10	B' and B'' are built only once	assumes $R \ll X$
DC flow	—	instant, linear, no convergence to worry about	no voltages, no reactive power

Measured convergence on the 3-bus example — largest error in $|V|$ and angle against the converged solution:

Method	1 iter.	2 iter.	3 iter.	5 iter.	Iterations for 4 decimals
Gauss-Seidel	0.0115	0.0022	0.0004	0.0000	5
Newton-Raphson	0.0013	0.0000	0.0000	0.0000	2
Decoupled	0.0087	0.0013	0.0001	0.0000	4
Fast decoupled	0.0089	0.0012	0.0002	0.0000	4

The program has **no convergence tolerance**: it runs exactly the number of iterations you ask for, and reports every one of them. If you want to watch the process, ask for several and read them one by one; if you only want the answer, ask for plenty.

Iterations are numbered from zero in the report: with `Iteraciones = 5` you will see blocks `k = 0` through `k = 4`.

9. Worked cases

All of them use the same system, the one loaded by **Home** ▶ **Cargar ejemplo de 3 barras** (load the 3-bus example):

Buses

#	Type	P	Q	$ V _{\text{esp}}$	V0	ang0
1	SL	—	—	1.05	—	0
2	PQ	−0.5	−0.2	—	1.00	0
3	PV	0.4	—	1.02	—	0

Lines

From	To	Data	R	X	Bsh
1	2	Z	0.02	0.06	0.06
1	3	Z	0.08	0.24	0.05
2	3	Z	0.06	0.18	0.04

Bus 2 is a load of 0.5 pu active and 0.2 pu reactive; bus 3 generates 0.4 pu while holding 1.02 pu of voltage.

The resulting admittance matrix, which heads every report:

```

1:  6.2500-18.6950i  -5.0000+15.0000i  -1.2500+3.7500i
2:  -5.0000+15.0000i   6.6667-19.9500i  -1.6667+5.0000i
3:  -1.2500+3.7500i  -1.6667+5.0000i   2.9167-8.7050i

```

Case 1 — Network from lines, Gauss-Seidel

1. **Home** ▶ **Gauss-Seidel**.
2. **Sistema**: **N de barras** (*bus count*) = **3** with ◀▶, **Ingreso de red** (*network input*) = **Líneas**, **Iteraciones** = **5**. Move down to **Datos de barras** (*bus data*) and press **Enter**.
3. **Barras**: on row 1, column **Tipo**, press **Enter** until it reads **SL**. Move right to **|V|esp** and type **1.05**, **Enter**.
 - Row 2: **Tipo** = **PQ**, **P** = **-0.5**, **Q** = **-0.2**, **V0** = **1**.
 - Row 3: **Tipo** = **PV**, **P** = **0.4**, **|V|esp** = **1.02**.
4. Go back with < and enter **Datos de red** (*network data*).
5. **Líneas**: press + three times and fill in the table above. Leave **Dato** as **Z**; **R** goes in **R/G** and **X** in **X/B**.
6. Press **Calcular** — the footer button, or the **Menu** key.

Expected result in the **k = 4** block, the last of the five iterations:

Bus	V	ang (rad)	Pcalc	Qcalc
1	1.0500	0.0000	0.1116	0.4164
2	1.0284	-0.0082	-0.5003	-0.1999
3	1.0200	0.0438	0.4000	-0.3430

Losses per line: (1-2) **0.0027-0.0567i**, (1-3) **0.0037-0.0425i**, (2-3) **0.0049-0.0274i**. **Network total**: **0.0112 - 0.1266i**.

How to read it: the PQ and PV buses reproduce their scheduled power almost exactly — the small remainder (**-0.5003** instead of **-0.5000**) is what is left to converge, and it disappears with one more iteration. The slack supplies **0.1112** pu once converged, which is the **0.1** of the balance plus the active losses. The negative imaginary part of the losses means the network **injects** net reactive power, from the capacitive line shunts.

Case 2 — The same system with Newton-Raphson

From the results, press < back to **Sistema**, change **Metodo** to **Newton-Raphson** with ◀▶ and **Iteraciones** to **3**, then press **Calcular**.

Nothing has to be re-entered. It converges to **the same solution** as Case 1, in 2 iterations:

Bus	V	ang (rad)
1	1.0500	0.0000
2	1.0284	-0.0082
3	1.0200	0.0438

Besides voltages and losses, every iteration reports:

- **Potencias y desbalances** (*powers and mismatches*) — **Pc**, **Qc**, **dP**, **dQ**
- **H reducida**, **N reducida**, **M reducida**, **L reducida** — the reduced blocks
- **Jacobiano reducido** — the complete system being solved
- **Residuos** (*corrections*) — **d-ang** and **d-V**, the updates applied

In **dP** and **dQ** you will see large values on the slack row and in **dQ** of the PV buses. That is normal: there is no unknown there, and those mismatches are not part of the system being solved. Watch **dP** on the non-slack buses and **dQ** on the PQ buses: those are the ones that must go to zero.

Case 3 — Decoupled and fast decoupled

Same as Case 2, changing only **Metodo**. With **4 iterations** both reach the same solution.

- **Desacoplado** (*decoupled*) solves $\Delta P = H \cdot \Delta \delta$ and $\Delta Q = L \cdot (\Delta V/V)$ with exact **H** and **L**, rebuilt every iteration. The voltage correction is labelled **d-V/V** because the unknown is the normalised one.
- **Desacoplado rapido** (*fast decoupled*) uses constant **B'** and **B''**, built once from the imaginary part of **Y**, and solves $\Delta P/V$ and $\Delta Q/V$. They show up in the report as **Bp reducida** and **Bpp reducida**.

Compare the blocks across iterations: in the decoupled method **H** and **L** change, in the fast one **B'** and **B''** are always the same.

Case 4 — DC power flow

Sistema ▶ **Metodo** = **Flujo DC** ▶ **Calcular**. The **Iteraciones** field is ignored: this is a direct method.

Expected result. The DC **B'** matrix:

20.8333	-16.6667	-4.1667
-16.6667	22.2222	-5.5556
-4.1667	-5.5556	9.7222

Bus	ang (rad)	ang (deg)	P (pu)	type
1	0.0000	0.0000	0.1000	SL
2	-0.0143	-0.8165	-0.5000	PQ
3	0.0330	1.8908	0.4000	PV

Line flows:

Line	X	P _{ab}	P _{ba}
(1-2)	0.0600	0.2375	-0.2375
(1-3)	0.2400	-0.1375	0.1375
(2-3)	0.1800	-0.2625	0.2625

How to read it:

- $|V| = 1.0000$ on every bus — that is a modelling assumption, not a result.
- The slack supplies 0.1000 pu, **exactly** what the balance needs ($0.5 - 0.4$). The DC model has no losses, hence $P_{ba} = -P_{ab}$.
- Compare with Case 1: in AC the slack supplied 0.1112 . The difference, 0.0112 , is precisely the active losses of the network. The DC flow gives you the active power split, not the losses.
- Resistances and the B_{sh} shunt **take no part** in the DC flow.

Case 5 — Entering the Y matrix directly

If the problem already hands you Y rather than the impedances:

1. **Sistema** ▶ Ingreso de red = **Matriz Y** ▶ Datos de red.
2. **Matriz Y**: the table already has one row per pair $i \leq j$. Fill in G and B for each. Elements below the diagonal fill themselves in.
3. **Calcular**.

All five methods work the same, but there will be **no per-line losses and no DC line flows**: the program warns "No disponibles: matriz Y directa" (not available: Y matrix entered directly).

Case 6 — Parallel lines and shunts

For two circuits in parallel between buses 1 and 2, add **two rows** with $De = 1$, $A = 2$ and the data of each. The program adds their series admittances and their B_{sh} before assembling Y , and the results show them as a single equivalent line (1-2).

About the shunt: B_{sh} is the **total** for the line. If the problem gives you $B/2 = 0.03$ per end, enter 0.06 . Quick check on the Y matrix that heads the results: the shunt appears added into the diagonal, half in $Y(1,1)$ and half in $Y(2,2)$.

Case 7 — Changing one input and recalculating

Any input can be changed and recomputed without redoing the rest:

- **Change a bus type**: on **Barras**, column **Tipo**, **Enter** cycles $SL \rightarrow PV \rightarrow PQ$. Marking a bus as **SL** turns the old slack into **PQ** by itself. Check afterwards which columns became editable or greyed out.
- **Change the bus count**: on **Sistema**, with ◀ ▶. The data you already entered is kept; new buses come in as PQ with $V0 = 1$. If you reduce the count, lines pointing at buses that no longer exist are clamped to the last valid bus, so review them.
- **Delete a line**: on **Líneas**, put the cursor on its row and press $-$.

- **Compare methods:** change only **Metodo** on **Sistema** and press **Calcular**. From the results screen itself, the **Menu** key recalculates.

10. Reading the results

The results screen is a scrollable report, 15 lines visible at a time. Move with ▲ ▼, or tap the upper or lower half of the screen to jump seven lines. If a line is wider than the screen, pan it with ◀ ▶: the report shifts 24 pixels per press.

Common blocks:

Block	What it is
Matriz de admitancia Y	the Y that was used, as $G+jB$
Voltajes / Estado actualizado	$ V $, angle and rectangular form
Potencias calculadas	the P and Q that follow from the current state
Perdidas en lineas	S_{ab} , S_{ba} and S_{loss} for each line, plus the total

Specific to Gauss-Seidel:

Block	What it is
P y Q usadas para actualizar V	the values that go into the V formula that iteration. On PQ buses they are the scheduled ones; on PV buses, Q is the recomputed one; on the slack, whatever the network gives

Specific to the Newton family:

Block	What it is
Potencias y desbalances	P_c , Q_c , dP , dQ
H N M L reducidas	Jacobian submatrices, without the slack (and without the PV buses where applicable)
Bp / Bpp reducidas	in the fast decoupled method, the constants that replace H and L
Jacobiano reducido	full NR only: the system being solved
Residuos	d-ang and d-V (or d-V/V in the decoupled method): the corrections applied

Specific to the DC flow:

Block	What it is
Matriz Bp de continua	built from $1/X$ of each line
Sistema reducido	B' without the slack row and column, and the P vector
Resultado	angles in rad and degrees, injections and bus type
Flujos por linea	X , P_{ab} and P_{ba} for each line

11. Warnings and errors

Message	What it means
Error: no se definio barra slack.	no bus is SL . Go to Barras and mark one
Aviso: Y(i,i) casi nulo.	that bus ended up isolated or with no useful connection; check the lines
No disponibles: matriz Y directa.	per-line losses need line data
Sin lineas para reportar.	you are in Líneas mode but did not add any
Linea (a-b) con X=0: se omite.	the DC flow needs reactance; a purely resistive line contributes nothing

If the answer looks wrong, check in this order:

1. The signs of **P** and **Q** ? Loads are **negative**.
2. Do the SL and PV buses have their **|V|esp** ? That is the voltage they are held at.
3. Is **Bsh** the total and not the half?
4. Is there exactly one slack bus?
5. Enough iterations? Compare against the table in section 8.
6. Is the network connected? A bus with no lines has no solution.

12. Limits

Buses	1 to 20
Iterations	1 to 50
Lines	no fixed limit
Report lines	3000 (anything beyond that is dropped)
Displayed precision	4 decimals

A large case with many iterations takes a while: the HP Prime solves the linear system in interpreted PPL. If the report gets cut off, lower the iteration count.

13. What is in this repository

File	What it is
Flujo de Potencia.hpprgm	the program. It is PPL text: you can open it in any editor
Flujo de Potencia.zip	the same file, compressed for transfer
FLUJO_HP.hpappdir/	an earlier version packaged as an HP Prime app, without touch support. Kept for reference
PRUEBA_HP.hpprgm	diagnostic program: it exercises each PPL construct the main program uses, once each, and displays key and touch codes. Handy if your firmware behaves differently
README.md / README.pdf	this document, in Spanish
README.en.md / README.en.pdf	this document
herramientas/generar_pdf.ps1	builds the PDFs from the .md files (section 15)

The **TI-Nspire** version lives in its own repository: github.com/SCLDarkKnight/Newton-Raphson-Gauss-Seidel-TI

14. Modifying the program

The source **is** the `.hpprgm`: plain text, editable, commented. Edit it, save it, and send it back with the Connectivity Kit.

Three things about PPL worth knowing before you touch it, also noted at the top of the file itself:

- **Never use `i` as a variable**: in PPL it is the imaginary unit. The program's indices are `ib`, `jb`, `kb`, `ia`, `ja`, `ka`.
- **A `LOCAL` statement takes at most 8 names**. With nine or more the calculator throws a *syntax error* on that line. That is why the declarations come in groups of six: if you add variables, open a new `LOCAL` line instead of extending an existing one.
- **Subroutines are forward-declared** at the top of the file. Without that declaration, a call is read as a global variable.

The key codes in use are listed in the file header (`Esc = 4`, `Enter = 30`, arrows `2 / 12 / 7 / 8` ...). If some key does not respond on your firmware, run `PRUEBA_HP` to see what code it emits and fix it wherever it appears.

The whole look comes from one place: the palette commented above the drawing section.

16. Licence

Released under the **GNU General Public License v3.0 (GPLv3)**. You may use, study, modify and redistribute it, as long as any derivative work keeps the same licence.

It is your duty to pass this program on to everyone it can help.