$$B^U = \{1 \le i \le n \mid 1 = x_i^L < x_i^U < +\infty\} \quad (1.3b)$$

$$B^R = \{1 \le i \le n \mid 1 < x_i^L < x_i^U < +\infty\} \quad (1.3c)$$

and refer to them as the set of *free variables*, the set of *upper bounded variables* and the set of *lower bounded variables* respectively. Fixed variables, if

`HessPhiProd()` computes the matrix-vector product of (2.9) with a vector.

At each inner iteration, a call to `PyGltr` solves the quadratic subproblem in the trust region $\|d\|_{k;j} \leq \Delta_{k;j}$ where $\Delta_{k;j} > 0$ is the trust-region radius and $\|\cdot\|_{k;j}$ is, by default, the ℓ_2-norm. The user may define a suitable preconditioner object and pass it over to `PyGltr` upon instantiation, e.g., using

```
G = pygltr.PyGltrContext( g, radius = TR.Delta, prec = M )
```

instead of

```
G = pygltr.PyGltrContext( g, radius = TR.Delta ).
```

In this case, $\|\cdot\|_{k;j} = \|\cdot\|_{M^{-1}}$, or, more generally, $\|\cdot\|_{k;j} = \|\cdot\|_{M_{k;j}^{-1}}$. In the above, `TR` is an instance of a `TrustRegion` object. Upon exit from `PyGltr`, steps are rejected if they are infeasible. In this case, the trust-region radius is reduced and the quadratic trust-region subproblem is solved again. Once the step is feasible, it is tested for acceptance based on the ratio of aceptanc22451 0 Td (hiev)Tj 17.7426 0 Td (ed)Tj 13.0727 0 whose purpose is to identify a strictly feasible initial point, may also be added, for instance based on `LSQP` or `FILTRANE` [GOT03a]. Rules to update the barrier parameters may be kadd4 T.BTf 5.tep27.8662 iladelphi(TR.Del6n,)Tj 17.991USATd (case4Tj 19.89720001 0 Td12264 T -334227 -1