lib

December 7, 2024

[32]:
```
def floatfmt(v, prec, exp):
    return f"{v/10**(exp):0=1.{prec}f}{f'e{exp}' if exp != 0 else ''}"
```

[36]:
```
class ValErr:
    val: float = 0
    err: float = 0
    err_set = False

    def __init__(self, val, err=0):
        self.val = val
        if err != 0:
            self.err_set = True
            self.err = err

    def getTuple(self):
        return (self.val, self.err)

    @classmethod
    def fromTuple(self, tup):
        return ValErr(tup[0], tup[1])

    @classmethod
    def fromFit(self, popt, pcov, i):
        return ValErr(popt[i], np.sqrt(pcov[i][i]))

    @classmethod
    def fromFitAll(self, popt, pcov):
        for i in range(0, len(popt)):
            yield ValErr(popt[i], np.sqrt(pcov[i][i]))

    def strfmt(self, prec=2):
        if self.err != 0:
            return fr"{self.val:.{prec}e} ± {self.err:.{prec}e}"
        else:
            return f"{self.val:.{prec}e}"

    def strfmtf(self, prec, exp):
```

```
        if self.err != 0:
            return fr"{floatfmt(self.val, prec, exp)} ± {floatfmt(self.err,␣
↪prec, exp)}"
        else:
            return f"{floatfmt(self.val, prec, exp)}"

    def strltx(self, prec=2):
        if self.err != 0:
            return fr"{self.val:.{prec}e} \pm {self.err:.{prec}e}"
        else:
            return f"{self.val}"

    def __repr__(self):
        return f"ValErr({self.val}, {self.err})"

    def __mul__(self, num):
        return ValErr(self.val * num, self.err * num)
```

[34]:
```
def spacearound(dat, add):
    return np.linspace(dat[0] - add, dat[len(dat)-1] + add)
```

[35]:
```
def div_with_err(a, a_err, b, b_err):
    err = (1 / b) * np.sqrt(a_err**2 + (a * b_err / b)**2)
    return (a / b, err)
```

[ ]:
```

```